Similarity Search in High-dimensional Spaces with Applications to Time Series Data Mining and Information Retrieval

Muhammad Marwan Muhammad Fuad

THESE / UNIVERSITE DE BRETAGNE SUD
sous le sceau de l'Université européenne de Bretagne

pour obtenir le titre de
DOCTEUR DE L'UNIVERSITE DE BRETAGNE SUD
Mention : Sciences et Technologies de l'Information et de la Communication

École Doctorale SICMA

présentée par
Muhammad Marwan Muhammad Fuad
Préparée au laboratoire de recherche en informatique(VALORIA)
N° d'ordre :

Similarity Search in High-dimensional Spaces with Applications to Time Series Data Mining and Information Retrieval

Thèse soutenue le 22 février 2011
devant le jury composé de :

Georges HEBRAIL
Pr. Telecom Paris Tech & Chercheur EDF R&D / *président*

Valérie MONBET
Pr. Université de Rennes1, IRMAR / *rapporteur*

René QUINIOU
Cr. INRIA, Bretagne Atlantique / *rapporteur*

Gildas MENIER
Mcf. Université de Bretagne Sud/ *Co-encadrant de thèse*

Pierre-François MARTEAU
Pr. Université de Bretagne Sud/ Directeur de thèse

Similarity Search in High-dimensional Spaces with Applications to Time Series Data Mining and Information Retrieval

A dissertation presented to the

Université de Bretagne Sud - Université Européenne de Bretagne

for the degree of

PhD in Information and Communication Sciences and Technologies

by

Muhammad Marwan Muhammad Fuad

February 2011

Beyond this place of wrath and tears
Looms but the horror of the shade,
And yet the menace of the years
Finds, and shall find, me unafraid.

Invictus-William Ernest Henley

To Marwan,
not the one I am;
but the one I wanted to be.

Acknowledgment

I would like to thank Prof. Pierre-François Marteau who accepted to supervise my PhD despite his many responsibilities. His broad knowledge and academic experience make of him a remarkable researcher.

I would also like to thank the chairman and the members of the dissertation committee.

Abstract

We present one of the main problems in information retrieval and data mining, which is the similarity search problem. We address this problem mainly from a metric perspective. We focus on time series data, but our general objective is to develop methods and algorithms that can be extended to other data types. We investigate new methods to handle the similarity search problem in high-dimensional spaces. The novel methods and algorithms we introduce are tested extensively and they show superiority over other methods and algorithms in the literature.

Contents

Chapter 1

Introduction

The similarity search problem is one of the fundamental problems in computer science. It has numerous applications in text, video and image retrieval, pattern recognition, bioinformatics, Web search, fingerprint databases, and many others. In this problem a pattern is given and the algorithm searches the database, or the Web, to return all or most, depending on whether the search is exact or approximate, of the data objects that are "close" to that pattern according to some semantics of closeness. In another variation of this problem, the algorithm may be asked to retrieve a specified number of the closest objects to that pattern. A naïve solution to this problem is to compare the pattern with each object in the database and return all the objects that are similar to that pattern. This similarity is depicted using a principal concept which is the similarity measure or its stronger form; the distance metric.

With the proliferation of less expensive storage units with an increasing storing capacity, the amount of data in modern databases is so large that efficiency and effectiveness have become the main issues in evaluating the performance of different similarity search algorithms.

Of the different paradigms proposed to manage the similarity search problem, the metric model stands out as one that is applicable to different data types. The distance metric on which the metric model is based is a strong mathematical tool which helped the researchers build different data structures specific to metric spaces. Other techniques, such as the pivot technique, are based on the triangle inequality, one of the axioms of distance metrics. All these advantages of the metric model made it a rich field of research in information retrieval.

Time series is a data type that is frequently encountered in several scientific, medical, and business applications. Although time series data can be handled using the different data structures that the metric model offers, the high-dimensionality of time series data, in addition to the large size of time series databases, usually require special approaches developed specifically for this type of data.

The main scheme that is widely used to handle time series data is the following: the high-dimensionality of these data is reduced by extracting lower-dimensional features of the time series. Then a similarity measure or a distance metric is defined on this lower-dimensional space. The above steps are performed at indexing-time. At query-time the query time series is projected on the same lower-dimensional space and all the time series, whose distance in the lower-dimensional space is smaller than a predefined real number known as *threshold*, are returned. The distance in the lower-dimensional space is chosen to underestimate the original distance defined on the raw

data. If not, the algorithm may return time series which are not true answers to the query in the original space, or the algorithm may miss some true answers to the query in the original space. In this case the search is said to be *approximate*.

The distance in the lower-dimensional space is chosen so that it is as close as possible to the original distance. An ideal distance on the lower-dimensional space would be equal to the distance in the original space. However, in most cases such an ideal distance is almost impossible to find, and the search algorithm returns a candidate answer set whose cardinality is larger than that of the true answer set, because this candidate answer set may contain false alarms which are time series that are close to the query in the lower-dimensional space, but they are farther than the threshold in the original space. The search process can handle these false alarms, as long as there are not many of them, through a post-processing linear scanning on the raw data using the original distance to filter out all the false alarms and recover the true answer set.

Some time series dimensionality reduction techniques use a similarity measure (not a distance metric) in the reduced space. In this case the model is non-metric because this measure may violate some of the axioms of the metric distance. This does not seem to affect the similarity search effectiveness since most of these similarity measures respect the triangle inequality. Anyway, other methods use distance metrics in the reduced space so the search is actually metric.

The scope of this dissertation is similarity search in its general sense. We focus mainly on time series data, but almost all our work can be extended to other data types. Our objective is to investigate novel techniques to solve the similarity search problem in high-dimensional data, mainly time series data. We also address the similarity search from a perspective that is as close to metric as possible. Although in all the conferences we attended the researchers we talked to were in disagreement on whether similarity search methods should focus on general models that can be applied to a wide range of data types, or if they should focus on developing domain-specific solutions that benefit from the particular nature of each data type. We do not claim to have found an answer to this question.

This dissertation is structured in the following manner:

Chapter 2-Similarity Search in Metric Spaces: This chapter introduces the similarity search problem as handled by the metric model. The chapter presents a formal introduction to the problem together with its applications. We also present the main different forms of the similarity search problem. We also present a mathematical definition of the distance metric on which the metric model is based and show some examples of it. Different strategies for addressing the similarity search problem using the metric model are presented in this chapter. Partitioning and pivots, two widely used techniques, are also presented in the chapter. The dimensionality curse is discussed and some of the techniques that are used to handle this problem, such as mapping methods, are introduced. The approximate similarity search methods and principles are only briefly discussed since they lie beyond the scope of this dissertation. In the second part of the first chapter we present some of the widely

used indexing structures in metric spaces. The non-metric model is mainly presented as an extension to the metric model to help the reader understand the limits of the metric model.

Chapters 2 and 3 are background chapters so they finish with a summary, unlike Chapters 4-6 in which we present the bulk of our work, so they finish with perspectives and future work.

Chapter 3-Time Series Data Mining and Information Retrieval: This chapter focuses on the similarity search problem in time series data. We start by giving a brief mathematical definition of time series and the translations that are frequent with them and how they are usually handled. We also present some of the main tasks of time series data mining. Then we introduce the general frame of handling time series data which we explained earlier and which is based on reducing the high-dimensionality of time series. We also give a description of the most widely-known dimensionality reduction techniques. SAX, which is probably the most important technique, is left to Chapter 5 where it is discussed in detail.

In the second part of this chapter we present the different similarity measures and distance metrics that are used in the time series community. We also present some of the most recent, state-of-the-art similarity measures and distance metrics applied to time series data and we show their advantages.

Chapter 4-Adaptive Sampling and Time Series Classification: In this short chapter we present the first contribution of our dissertation. In this chapter we present our preliminary work published in [122] which is an experimental study that evaluates the impact of dimensionality reduction on a time series classification task. The experiments presented are based on a representation method which reduces the dimensionality by using an adaptive sampling technique of the time series. The experiments presented in this chapter utilize different compression ratios and different similarity measures and metric distances.

The reason we present this chapter, although the work introduced is preliminary, is that it is a part of the work of the dissertation. The second reason is that we realized recently how this preliminary work can be exploited as future work to improve the MIR method we present in Chapter 5.

We meant for all our chapters to be self-contained so we did not merge this short chapter with any other because it does not fit in anywhere else.

To avoid repetition, our papers have been cited only once at the beginning of Chapters 4, 5, 6 and were not cited again in these chapters

Chapter 5-Symbolic Methods in Time Series Information Retrieval: This is the chapter where we present the first major direction of our contributions. These contributions concern symbolic representation and symbolic distances and similarity measures.

The edit distance, which is a metric distance, is the main distance used to compare two strings. It is the reference distance in many applications in text mining and bioinformatics. However, this distance has its limitations because it considers local similarity only. We show by examples how this distance is unable to detect similarities between strings that are intuitively similar.

The first principal direction of contributions presented in this chapter is several versions of different improvements of the edit distance which add a global feature of similarity in addition to the local one that the edit distance considers, all by keeping the same complexity as that of the edit distance. We also show that all these versions are metric. We present extensive experiments to show the advantages of our improved versions of the edit distance, and to investigate how these improvements can be developed to handle other problems in time series data mining. The work we presented in this part of Chapter 5 was published in [129], [133], [134], [135], and [137]. We also present the results of another paper that was accepted in IADIS Multi Conference on Computer Science and Information Systems 2008, Amsterdam, Netherlands, 22–27 July 2008 (http://www.mccsis.org/2008/) but we decided at that time not to publish that paper because we wanted to develop this version (PFEED) further and to present it in another context.

The experiments on all the improved versions we present in this chapter are conducted in the context of time series data because it is our field of research. But we also refer to another paper [14] presented by another author who used one of our proposed distances in another context and obtained good results.

The proof of Lemma 1 in this chapter (which we present in the Appendix) is the proof we presented in the published papers, which is the reason why we are presenting it in this dissertation. However, after presenting that lengthy proof we realized a more compact proof can be obtained as a special case of the proof of Theorem 3.

The second principal direction of contributions in Chapter 5 concerns SAX. SAX is probably the most competitive symbolic representation method in time series information retrieval. The main advantage of SAX is that the similarity measure it uses in the lower-dimensional space is fast to calculate because it is based on pre-computed distances obtained from look-up tables. Our second principal contribution in Chapter 5 is a new similarity measure which is also fast to compute because it uses pre-computed distances too. But our new similarity measure is tighter and more intuitive than the original one. This part of Chapter 5 was published in [128].

Chapter 6-Multi-resolution Approaches to Time Series Indexing and Retrieval: Multi-resolution methods which use several lower-dimensional spaces have already been used in multimedia retrieval. In this chapter we introduce another major contribution of this thesis which is new multi-resolution time series indexing and retrieval approaches that we presented in [130], [132], [131], and [136].

In order to reduce the number of query-time distance evaluations and thus speed up the similarity search, our multi-resolution approaches are used to economize query-

time distance evaluations by using different resolution levels. Fast-and-dirty filters, which are based on the triangle inequality, are used to filter out the time series which are not answers to the query by utilizing stored, pre-computed distances.

The first of our three multi-resolution methods is “weak MIR”. This method is a standalone time series retrieval method that uses two filters. The second method is “strong MIR” which uses one filter only, together with the lower-dimensional distance of a time series dimensionality reduction technique. Our last multi-resolution method is “Tight_MIR”, which has the advantages of the two previously mentioned methods. All these versions are validated through extensive experiments.

It is important to mention that the great majority of our experiments in this dissertation were conducted on time series datasets from [190]. The data sets in this archive have been widely-used for years. So testing on these datasets ensures that the collection represents the interest of the data mining/database community, and not just one group [57].

Chapter 7-Conclusion and Future Work: In this concluding chapter we summarize our contributions and present different directions of future work.

Chapter 2

Similarity Search in Metric Spaces

This is a background chapter on similarity search in metric spaces where we present fundamentals of this problem from a metric space perspective. In Section 2.2 we present some applications of this problem in different fields of computer science, and then we give a formal introduction of metric spaces in Section 2.3. These spaces are based on the concept of metric distance which is presented in Section 2.4. The different forms of the similarity search problem are presented in Section 2.5. A panorama of the different solutions to this problem is presented in Section 2.6. The general model of embedding in lower-dimensional spaces and other related issues are presented in Section 2.7. Approximate search, although not the topic of this dissertation but a principal technique of addressing the similarity search problem, is presented in Section 2.8. In Section 2.9 we introduce multimedia indexing structures. Metric spaces have the advantage of using certain data structures. A survey of these structures is presented in Section 2.10. A unifying model of the similarity search problem in metric spaces is explained briefly in Section 2.11. The non-metric model, a rapidly growing field of similarity search, is presented in Section 2.12, and we conclude this chapter with a summary in Section 2.13.

2.1 Introduction

Similarity search is a fundamental problem in computer science. This problem has many applications in multimedia databases, bioinformatics, pattern recognition, text mining, computer vision, medicine, data mining, machine learning and so on. With the advent of the internet and the increasing use of it, this problem has received more attention from researchers. The wealth of information available on the internet could not be manageable if search engines were not present. It does not make much sense to know that a piece of information is out there had we not got access to it. The usefulness of information depends highly not only on its quality, but also on the speed at which it is retrieved, which, in turn, depends upon the way it is represented and indexed. This all raises questions on indexing, representation and retrieval methods. Small databases that contain simple data objects can be handled easily. But managing large databases, like many of the databases in use today, requires serious effort, especially when they contain complex data types.

Traditional databases have been managed by using exact search queries, i.e. the search algorithm tries to retrieve the data objects that are identical to a given query. This was mainly the case with structured databases that contained numerical or alphabetical data. The approaches used to deal with these data bases were efficient.

But later substantial advances in the field of databases took place; the size of these databases grew enormously and is still growing that what was considered a large database just few years ago is now considered small. The data types themselves changed dramatically in nature. Modern databases contain data types such as images, audios, videos, time series, fingerprints, documents, DNA and protein sequences, and new data types do not stop emerging all the time. Another substantial change is the type of queries posed, which is, somehow, related to the two latter changes; searching for data objects that are identical to a given query in unstructured, weakly-structured, or imprecise data bases, like many databases in use today, may not be very meaningful, not to mention that in many cases the user may not be sure of what they are looking for when they pose the initial query. Hence *range query*, in which the user is interested in retrieving all the data objects that are within a predefined threshold of a given query, or *k-nearest neighbor*, in which the user tries to retrieve the k closest data objects to the query, have become popular. All these problems make traditional retrieving techniques inadequate that it is inevitable to think of new ways to handle these databases.

Before we introduce the problem formally and discuss the different solution paradigms, we start by presenting some of its numerous applications in different fields of computer science.

2.2 Applications of Similarity Search

There are quite a few applications of the similarity search problem. Here are some of them [40]. Our purpose is not to list all the applications of this problem but rather to provide a breadth of examples so that the reader can understand the different contexts in which this problem may be encountered.

2.2.1 Queries by Content

Multimedia content queries have received a lot of attention lately due to the continuous growth of user generated content [2]. Query Multimedia databases contain data types that can not be ordered, like images, fingerprints, video or audio. Queries in these databases do not search for exact matches, simply because the query itself may not be precise: the probability that two images match pixel-wise is almost zero, unless they are digital copies of the same source, also in music information retrieval musicians play the melody differently, so searching for exact matches is unlikely to make much sense. In principle, in music information retrieval one can extract various acoustic features and use distance function dependent on these features [56].

In multimedia databases queries by content can be searched in several ways such as free text query, query by textual descriptor [118], or by query-by-example (QBE)

[218] where the user provides a part of the document as a query and the algorithm searches for similar documents.

2.2.2 Text Mining

Textual documents are not structured in many cases. Many of them contain spelling or *optical character recognition* (OCR) errors. The Web is a good example of such documents. All this causes problems similar to those encountered in multimedia databases. Moreover, these textual databases may not only be searched for strings, but also for semantic concepts. This type of search can not be handled using classical approaches. The World Wide Web, for instance, has a large size and other special features that can not be dealt with the same way we deal with classical corpora [169].

Text mining tackles the general data mining tasks such as classification [81], clustering [100], representation [90], and anomaly detection [6], in addition to other specific text mining tasks like spam filtering [70].

2.2.3 Computational Biology

Computational biology consists of the development and use of mathematics and computer science technologies to help solve molecular biology problems [170].

Three of the four data emerging from a cell are text data; DNA, RNA, and protein sequences [1]. These data can be handled as strings and there has been successful research on how to benefit from textual database algorithms to manage DNA and protein sequences the same way we manage texts. A typical problem that we encounter in computational biology is to search for a short pattern (a series of amino-acids, for instance) in a larger sequence. Here, again, the search may not be exact due to the fact that there are genetic differences, so similarity search presents important appropriate algorithms and solutions to deal with many problems in computational biology. In fact, many similarity measures that are widely used have been introduced by computational biologists.

There are several sequence similarity algorithms that are used in computational biology like *Basic Local Alignment Search Tool* (BLAST) [8] which is based on identifying local alignments between sequences by finding short matches and from these initial matches (local) alignments are created [51]. BLAST was in fact a development of the Smith-Waterman algorithm [177] which allows consideration of insertions/deletions and compares fragments of arbitrary lengths between two sequences thus identifying the optimal local alignments.

2.2.4 Pattern Recognition

Pattern recognition, also named "learning from examples", often involves the definition of stochastic models, like neural networks or Hidden Markov Models (HMM), which are trained on training data and tested on unseen test data [185]. Pattern recognition classifies objects into a number of classes or categories based on the patterns that the objects exhibit [186]. This classification task can be seen as a binary-class problem where the object belongs or not to a particular class or a multi-class problem where learning concerns several classes [139].

2.2.5 Audio and Video Compression

Audio data have different sources of redundancy which permits compressing it. The main source of redundancy is that adjacent audio samples tend to be similar, so we can simply subtract each audio sample from its predecessor and encode the differences [163].

In internet-based audio or video conferences transmitting over narrow-band channels is a big problem. In this context, a frame (a static picture of a video or a fragment of an audio) can be thought of as consisting of a number of sub-frames (possibly overlapping) and the problem can be solved by sending the first frame as it. As for the next frames, they are constructed by sending only the sub-frames that are significantly different from the previously sent ones [40]. □

The above applications are based on similarity, which is an intuitive concept. Similarity search can be viewed as retrieving all the data objects, in the repository, that are "near" a given query. This nearness can be modeled using a powerful mathematical concept; the *metric distance,* which is related to another mathematical concept; the *metric space*.

2.3 The Metric Space

Let D be a set of objects. A function d

$d : D \times D \to \mathsf{R}^+ \cup \{0\}$, 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) $d(x,z) \leq d(x,y) + d(y,z)$ (triangle inequality)

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

There are other variations of this form which satisfy weaker or stronger conditions [58].

Search in metric spaces has many advantages, the most famous of which is that a single indexing structure can be applied to several kinds of queries and data types that are so different in nature. This is mainly important in establishing unifying models for the search problem that are independent of the data type,. This makes metric spaces a solid structure that is able to deal with several data types [214].

In metric spaces the only operation that can be performed on data objects is computing the distance between any two objects, which enables us to determine the relative location of the data objects to one another. This is different from the case of vector spaces; a special case of metric spaces, where data objects have k real-valued coordinates which makes it possible to perform operations that can not be performed in general metric spaces, like addition or subtraction, for instance. Vectors have certain geometric properties that can be exploited to construct indexing structures, but these properties can not be extended to general metric spaces [38].

The extensive applications of vector spaces may raise a question on the importance of working with metric spaces which have limited exploitable geometric properties. But in fact, the numerous applications of the vector spaces do not lessen the importance of the research done on metric space-oriented methods. There are many cases that can not be managed by any kind of vector-like structure. Take the case of strings, for instance, they are widely used in the textual databases or bioinformatics communities, yet they can not be represented as vectors. There are also other cases where the data objects are represented in vector spaces, still the nature of the problem makes it easier to handle them in metric spaces, like when searching for images using color similarity. In this case there is *cross-talk*, i.e. correlations between vectors. This cross talk is taken into consideration by user-defined weights using distance functions [214].

Another weakness of vector spaces is the so called *dimensionality curse*; an important problem that we will explain in more details later. This problem can slow down the search to make it similar to that of *linear scanning* (sometimes called *sequential scanning*), or it makes the search index consume too much space. On the other hand, some data structures use vectors that have very limited dimensionality (binary, for example) that using these coordinates does not provide much help [38].

But may be the most important argument in favor of metric spaces is that they provide solutions that are highly extensible in that they are not only applicable to many data types that are in existence today, but also to other types that may exist in the future [214].

2.4 The Distance Functions

There are so many distance functions that are known in the multimedia community, some of them are general, while others are used with certain types of data structures. In the following, we present some of the most widely utilized distance functions.

The Minkowski Distance: This is actually a whole family of distances, designated by L_P .This distance is defined in n -dimensional space as:

$$L_P[(x_1, x_2,...,x_n),(y_1, y_2,..y_n)] = \sqrt[P]{\sum_{i=1}^{n} |x_i - y_i|^p} \qquad (2\text{-}1)$$

In Minkowski distance p does not have to be an integer, but it should not be less than 1. If $p = 1$, the distance is called the *Manhattan distance* or the *city block distance*. If $p = \infty$, it is called the *infinity distance* or the *chessboard distance*. And if $p = 2$,we get the well-known *Euclidean distance.*. Figure 2.1 shows a few examples of the Minkowski distance. A variation of these distances are the *weighted Euclidean distance* or the *weighted maximum distance*, where additional weights: $w_1, w_2, ..., w_n$ are assigned to the dimensions.

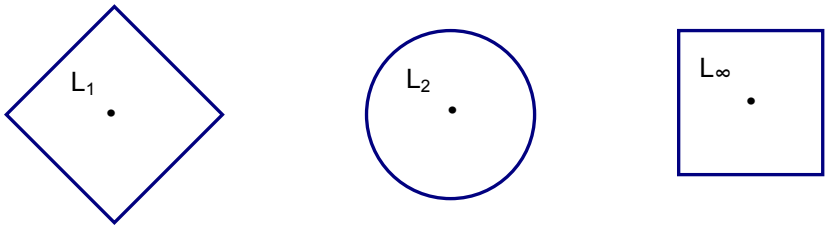


Fig. 2.1. Minkowski distance for L_1, L_2, and L_∞

Notice: In (2-1) if $p \in [0,1]$ the distance is called the *fractional distance* [3] which is a non-metric distance because it violates the triangle inequality.

The Hamming Distance: The Hamming distance between two strings S , T , denoted $H(S,T)$ is defined as [98] :

$$H(S,T) = |\{i : s_i \neq t_i\}| \qquad (2\text{-}2)$$

The Hamming distance considers only the mismatches between two strings. If we consider other atomic operations, we get the *edit distance*

The Edit Distance: The edit distance [110], also called the *Levenshtein distance,* is a distance between two strings S, T , and is defined as the minimum number of delete, insert, and change operations needed to transform string S into string T . This distance is the main distance measure used to compare two strings. In [199] an algorithm was presented to solve this problem in time proportional to the product of the lengths of the two strings. The edit distance is mainly used in automatic spelling correction. This distance will be discussed further in later chapters.

The Multi-set Distance: Given a string S . Let $ms(S)$ be the multi-set, also called *bag* of symbols, that S contains. For example, if $S = '$*multimedia* $'$ then $ms(S) = \{m,u,l,t,i,m,e,d,i,a\}$ .Then the multi-set distance between S , and T can be defined as

$$d_{ms}(S,T) = max\{|ms(S) - ms(T)|, |ms(T) - ms(S)|\} \quad (2\text{-}3)$$

where the difference here has a bag semantics, e.g. $\{b,a,b,c\} - \{a,a,c,c\} = \{b,b\}$, and where $|\ .\ |$ denotes the number of elements in the bag.

It can easily be proven that this is a distance metric. [214]. It is interesting to notice that this distance is a lower bound of the edit distance, i.e.

$$d_{ms}((S,T) \le d_{edit}(S,T) \qquad \forall S,T \in \Sigma^* \quad (2\text{-}4)$$

2.5 Similarity Queries

A similarity query is defined as a pattern or query object, which does not necessarily belong to the database, and a constraint that determines the extent of proximity that the data objects should satisfy to qualify as answers to that query.

In information retrieval there are very well-known types of queries and others that are less known. Here are a few of these queries starting with the most widely known ones.

2.5.1 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 of that query. This can be represented as:

$$R(q,r) = \{u \in U; d(q,u) \leq r\} \tag{2-5}$$

where U is the set of objects in the database

Figure 2.2 gives an example of a range query

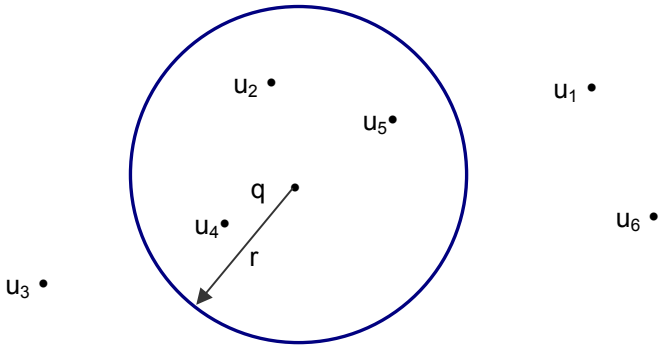


Fig. 2.2. Range query $R(q,r)$

Range queries have a main drawback; in some cases we do not have any prior knowledge about the database in question, so assigning an inappropriate value to r may sometimes result in two undesirable situations; returning a response set that is too large, or returning an empty response set. What happens in these cases is that the user restarts the query using a different value of r, and this may happen several times before getting a satisfying size of the response set. In very large databases, with a computationally expensive distance function, this can be tedious.

2.5.2 k – Nearest Neighbor Queries

In this kind of queries we look for the most similar, i.e. the closest, object in the data base to a given query. In the general case we look for the k most similar objects. Unlike the case with range queries, the response set here is never empty. Moreover, its size is defined beforehand by the user. Formally, this problem can be defined as:

$$kNN(q) = \{X \subseteq U, |U| = k \quad \wedge \forall u \in X, v \in U - X : d(u,q) \leq d(v,q)\} \tag{2-6}$$

In the case where several objects lie at the same distance from the query, ties are broken arbitrarily.

It is worth mentioning that range queries are most frequent in data warehousing, while nearest neighbor queries are most frequent in multi-media systems [18], [19].

Notice also that the nearest neighbor is not a symmetric relation, so if point u_1 is the nearest neighbor of u_2 this does not necessarily mean that u_2 is the nearest neighbor of u_1.

2.5.3 k – Reverse Nearest Neighbor Queries

In the past few years several researchers [105], [179], [180], [209] have studied a new similarity search problem which is the reverse nearest neighbor problem. The formal definition of this problem is as follows:

$$kRNN(q) = \{X \subseteq U, \forall u \in X : q \in kNN(u) \wedge \forall u \in U - X : q \notin kNN(x)\} \quad (2\text{-}7)$$

In simple words, in this kind of queries we are interested in retrieving the data objects that have q among their $k-$th nearest neighbor. This problem has many applications in mobile devices, data streaming, bioinformatics, and document databases. An important property of this problem is that data objects may qualify as answers even if they are far from the query, and data objects that are close to the query may not qualify. This is related to the density of the data objects in the data base.

2.5.4 Similarity Join

This problem is widely encountered in *data cleaning* or *integration*, where we have to provide error-free data. The similarity join between two sets A, B is the set of all pairs of data objects whose distance does not exceed a given threshold. This can formally be defined as:

$$J(A, B, r) = \{(a, b) \in A \times B : d(a, b) \leq r\} \quad (2\text{-}8)$$

A special case of this problem is the similarity self join, where $A = B$.

The similarity join problem requires more distance computations than the $k-$ nearest neighbor problem, and much more computations than the range query problem. In fact, despite the numerous applications of this problem, its quadratic computational complexity prevents from applying it to large data collections [59].

2.6 Strategies for Executing Similarity Queries

2.6.1 Complexity Considerations

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 that it needs. Different distances also need different computing times, the multi-set distance, for example, is faster to compute than the edit distance.

2.6.2 Sequential Scanning

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

The $kNN(q)$ query is performed in a similar way; an initial set P containing k arbitrary objects of the data base is formed. These k objects are ordered according to their distance from q . Then the execution continues with the other objects of the data base. So each object $u_i \in U - P$ is considered in the response set if and only if $d(u_i, q) < d(u_k, q)$, where u_k is the $k-$th nearest neighbor of q at a given stage. Whenever a new object is inserted in the response set the $k-$th object is eliminated. We continue until all elements of $U - P$ have been examined. □

It is easy to notice that in the cases where the size of the data base is very large, like most data bases in use today, sequential scanning is not the best scenario to answer proximity queries, because it requires so many distance evaluations.

Despite the inefficiency of sequential scanning it is used as a reality check of many complex indexing structures or high dimensional data [79].

Alternative algorithms to sequential scanning focus on reducing distance calculations as much as possible. This is achieved by using indexing structures, which are offline procedures based on storing some distance calculations. Later, and at query time, these calculations can be used to exclude some data objects, 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 data objects is scanned sequentially against the query to get the true answers of the query. □

Indexing structures can generally be divided into pivot-based indexing structures and compact-partitioning indexing structures.

2.6.3 Pivot Techniques

A few methods have been proposed to limit the number of distance calculations. One of these methods is the use of pivots [29]. The pivot technique uses a set of k pivots $\{p_1, p_2, ..., p_k\}, p_i \in U$ which are distinguished points, usually selected from the points of the database. By using the triangle inequality it follows that for any $u \in U$ we have:

$$d(p_i, x) \leq d(p_i, q) + d(q, x) \tag{2-9}$$

We also have:

$$d(p_i, q) \leq d(p_i, x) + d(x, q) \tag{2-10}$$

For the two inequalities together we have a lower bound on $d(q, x)$ which is:

$$d(q, x) \geq \left| d(p_i, x) - d(p_i, q) \right| \tag{2-11}$$

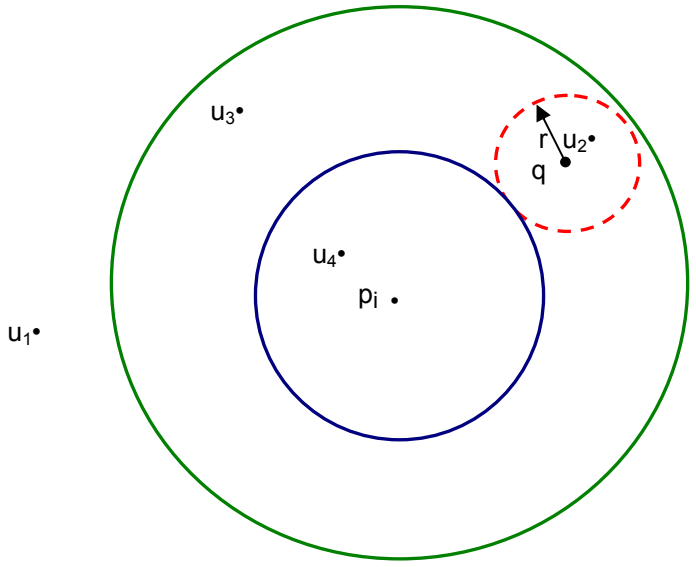


Fig. 2.3. Range query $R(q, r)$

With the use of pivots the search is "shifted" to be centered on these pivots. So using these pivots serves as a fast-and-dirty filter that excludes those objects that can not be answers to this query. This is performed by using the following condition, which is called the *exclusion condition*:

$$\left|d(p_i,u)-d(p_i,q)\right|>r \tag{2-12}$$

for some pivot p_i , and where $u \in U$

The above inequality excludes those objects that are not matches, but it does not guarantee reporting matches only. In other words, it guarantees no false dismissals, but does not guarantee no false alarms. These are two side-effects of similarity search algorithms that will be discussed further later.

In order to apply the exclusion condition we have to compute, at query time, the distances $d(p_i,q)$. This is what we call the *internal complexity* of the algorithm. Then in order to get the real matches, we have to sequentially scan the object candidate list, this is what we call the *external complexity* of the algorithm.

We see in Figure 2.3 that the exclusion condition excluded the objects u_1 , and u_4, and this is desired, since u_1 is not a match, neither is u_4 . It also kept u_2, and this is also desired, since it is a match. However, it failed to exclude u_3, even though it is not a match. This shows that the exclusion condition may cause false alarms. It only saves computing time, in that by using this exclusion condition we do not have to do unnecessary computations between q and objects in the data base that can not be matches.

2.6.4 Compact-partitioning Techniques

The aim of this partitioning is to divide the search space into sub-spaces so that the query is searched in some of these sub-spaces only, instead of accessing the whole search space to answer the query.

Ball Partitioning: This is the simplest type of partitioning. Ball partitioning [191] divides the search space U into two subsets U_1 and U_2 by using a pivot p as a reference point. The algorithm starts by choosing the pivot arbitrarily, then the distance between the pivot and all the data objects in U if computed. The data objects are divided between the two sets U_1, U_2 according to the following rules:

- If $\{d(u_i, p) \le d_m\} \rightarrow$ assign u_i to U_1
- If $\{d(u_i, p) \ge d_m\} \rightarrow$ assign u_i to U_2

where d_m is the median of all distances $d(u_i, p)$, $\forall u_i \in U$.

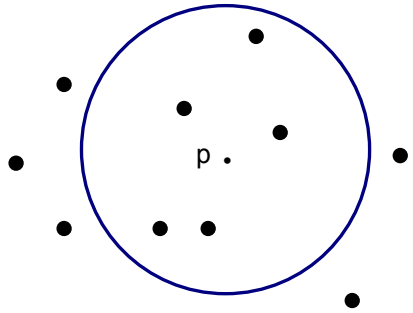


Fig. 2.4. Ball partitioning

The equality in the conditions $\leq, \geq$ is meant to assure balance partitioning in the case when the median is not unique. Figure 2-4 shows an example of ball partitioning.

Generalized Hyperplane Partitioning: This was also introduced in [191]. This type of partitioning can be viewed as an orthogonal ball partitioning. In Generalized hyperplane partitioning U is divided into two subsets U_1 and U_2 by using two pivots instead of one. The two pivots are also chosen arbitrarily and all the other data objects are assigned to either of the two pivots according to the following rules:

- If $\{d(u_i, p_1) \leq d(u_i, p_2)\} \rightarrow$ assign u_i to U_1
- If $\{d(u_i, p_1) \geq d(u_i, p_2)\} \rightarrow$ assign u_i to U_2

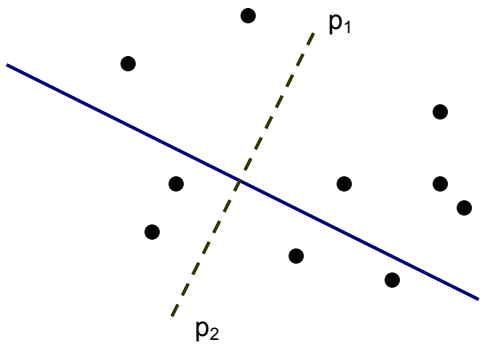


Fig. 2.5. Generalized hyperplane partitioning

Unlike ball partitioning, generalized hyperplane partitioning does not guarantee balanced partitioning. Figure 2-5 shows an example of generalized hyperplane partitioning.

Excluded Middle Partitioning: This partitioning [211] divides U into three subsets; U_1, U_2, U_3 according to the following rules:

- If $\{d(u_i, p) \leq d_m - \rho\}$ $\rightarrow$ assign u_i to U_1
- If $\{d(u_i, p) > d_m + \rho\}$ $\rightarrow$ assign u_i to U_2
- Otherwise $\rightarrow$ assign u_i to U_3

where 2ρ is the thickness of the excluding zone.

The motivation behind this partitioning is that whenever the query lies within the partitioning threshold, the search enters the two subsets making the algorithm degrades to the case of sequential scanning. So this partitioning leaves out an excluding zone making the search exclude one subset, at least. Figure 2.6 gives an example of excluded middle partitioning.

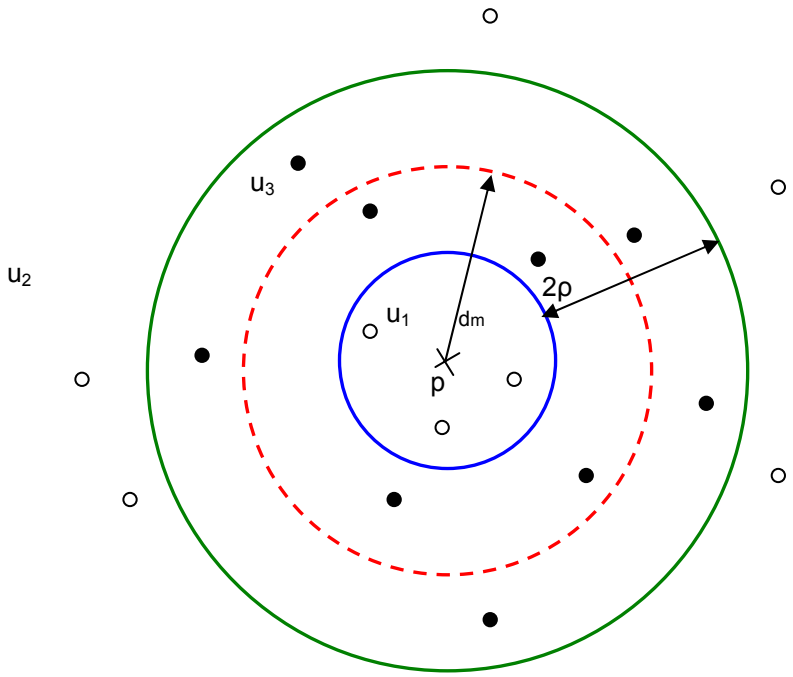


Fig. 2.6. Excluded middle partitioning

The previous types of partitioning can be generalized in two main ways; the binary partitioning can be extended to multiple partitioning, and the partitioning process itself can continue recursively in a top-down way to build a tree.

2.6.5 The Choice of Pivots

The way pivots are chosen has been the subject of much research in information retrieval, whether they are used for elimination based on pre-computed distances or as reference points for partitioning structures. Although many pivot-based algorithms simply choose them at random, it is well-known that this choice can affect the performance of search algorithms [138], [37]. It is easy to understand that the more data objects lie close to a certain pivot, the more the chance is that the query will lie in the vicinity of that pivot. Many researchers have worked on finding a distance distribution to determine the density of data objects in the search space [20], [63], [49], and others. But such a distribution is hard to find and harder to manage. This is one of the main reasons that the random choice of pivots is very frequent in many algorithms. Another reason why this choice is made randomly is because, surprisingly, in many cases the results obtained with randomly chosen pivots are quite acceptable [214].

Nevertheless, it is clear that some points can make better pivots than others. This can be illustrated by Figure 2.7: the two pivots have the same covering radius, but the distance between them is not the same. In case (a) the overlap area between the two ball partitions is larger than in case (b) so there are probably more data objects in case (a) than in case (b). As a result, the filtering efficiency of pivots p_1, p_2 is in case (a) less than it is in case (b) because whenever the search reaches an object that lies in the overlap areas it should access the two regions, which is an undesirable situation. This example helps us understand the well-known recommendation that the best choice for pivots is outliers; the objects that lie as far as possible from the other data objects in the search space. It is important to remember, however, that in generic metric spaces, and because of the properties of these spaces, such a task can be very time-consuming.

The point here is that since there is no rule about the number of pivots to be used, and since storing pivots and their corresponding distances (the distances between them and all the data objects in the search space) requires sufficient memory, a small set of well chosen pivots can be as effective as another large, randomly-chosen set of pivots. On the other hand, a well chosen set of pivots is more effective than a randomly chosen set of pivots of the same size.

A few heuristics have been proposed to select better sets of pivots. [138] proposes selecting as new pivots the objects that maximize the sum of distances between the new pivot and the pivots previously selected. Other heuristics have been proposed too [210]. But most of these heuristics do not work in generic metric spaces.

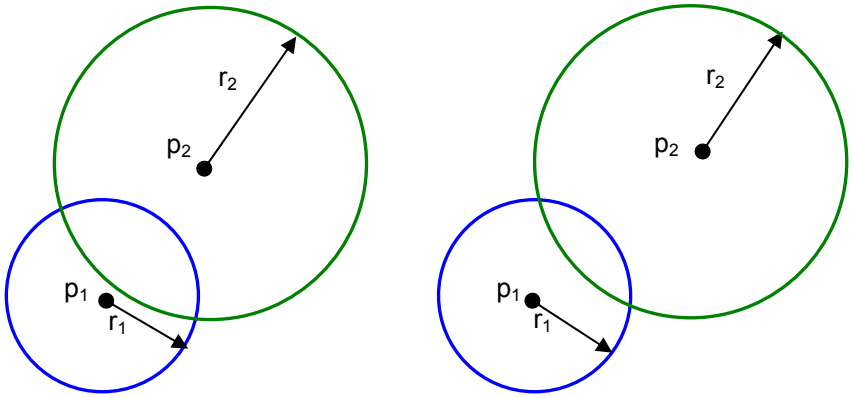


Fig. 2.7. The filtering efficiency of pivots is related to the distance between them

In [29] the authors present an extensive study of this problem and they present an *efficiency criterion* to compare two pivot sets. Their criterion states that of two sets of pivots of size k : $P = \{p_1, p_2, ..., p_k\}, P' = \{p_1', p_2', ..., p_k'\}$, P is better than P' when :

$$\mu_{D\{p_1, p_2, ..., p_k\}} > \mu_{D\{p_1', p_2', ..., p_k'\}} \qquad (2\text{-}13)$$

where μ_D is the distance distribution of the search space.

The bottom line about the choice of pivots is that they should be far away from other objects in the search space and far away from each other.

2.7 Embedding Methods

One of the strategies that we can use to perform less expensive distance calculations is to embed all the data objects of the metric space into another metric space. This may include using a different distance function. Embedding is also used in other fields of computer science, such as computer vision or computational biology, where the data are high-dimensional and complex, to map data sets into simpler and more compact representations [83]. The similarity query itself is transformed to the new space and performed there under certain restrictions to guarantee that the results obtained in the new space are relevant to those in the original space.

When embedding a metric space into another and performing the query in the transformed space, two main side-effects may be encountered; *false alarms*, also called *false positivity,* and *false dismissals*. False alarms are data objects 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 data objects 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 scanning is usually performed on the results of the query in the transformed space to filter out these data objects. However, false alarms can slow down the search time if there are too many of them.

False dismissals are a more serious problem and they need more sophisticated procedures to be avoided. Nevertheless, their influence on the similarity search is highly related to the nature of the query. If a user is selecting an item from an online store and he is looking for all the items that are similar to that item, then false dismissals are quite acceptable. But if the search query is performed to look for matches of a fingerprint or a DNA of a criminal, then false dismissals are not tolerated at all.

False alarms and false dismissals are dependent on the transformation used in the embedding. If f is a transformation from a metric space $(S_{original}, d_{original})$ into another space $(S_{transformed}, d_{transformed})$ then in order to guarantee no false dismissals this transform should satisfy:

$$d_{transformed}(f(u_1), f(u_2)) \leq d_{original}(u_1, u_2), \quad \forall u_1, u_2 \in S_{original} \qquad (2\text{-}14)$$

The above condition is known as the *lower-bounding lemma.* [4].

If a transformation can make the two above distances equal for all pairs of data objects in the original space, then similarity search produces no false alarms or false dismissals. Unfortunately, such an ideal transformation is very hard to find. Yet, we try to make the above distances as equal as possible. In other words, (2-14) can be rewritten as:

$$0 \leq \frac{d_{transformed}(f(u_1), f(u_2))}{d_{original}(u_1, u_2)} \leq 1 \qquad (2\text{-}15)$$

A *tight* transformation is one that makes the above condition as close as possible to 1.

Embedding can be achieved in different ways. One of them is to map the data objects in the original space into a vector space of a lower dimensionality. The distances in the transformed space are usually chosen from the L_P family, since these distances are generally inexpensive to compute. Similarity queries are performed by mapping the query q into the transformed space using the function f then the query is performed in this transformed space using these inexpensive distances. The response set obtained is post-processed to filter out the false alarms.

2.8 Approximate Similarity Search

Approximate similarity search is a broad topic in information retrieval. The motivations behind it are numerous; many exact similarity search methods are time-consuming, that in some cases the response time becomes unacceptable. Besides, in many applications, the overhead time necessary to achieve exact search is not worth the importance of the results obtained. If a user is looking for an image that he wants to download to use as a desktop background then it is not important to use an exact algorithm capable of retrieving all the images in the data base that are similar to the image that he is looking for, because such a thing would probably take a long time that the search process would be meaningless. In fact, this example presents another motivation of approximate similarity search; in many cases when the user launches the query, he may not have a clear idea of what he is looking for, and only after getting an approximate response set does he decide to launch another, more precise query.

[66] classifies approximate similarity search methods into two main categories:

- Methods that exploit the transformations of the metric space.
- Methods that reduce the size of the data objects being examined.

A more recent classification of approximate similarity search methods appears in [151] and [48]. In this classification we also have two classes of methods:

- **Changing Space:** These methods depend on first changing the metric space, which can be achieved either by changing the distance measure or the object space, then on solving the exact problem on the obtained space where the search is simpler. An example of these methods is an approximate method VA-LOW [201], which is based on the VA-file [202]. The VA-file is a sequential structure which contains approximations of spatial objects based on a fixed number of bits.

- **Reducing Comparisons:** These methods use the exact distance, but they speed up the search by limiting the number of objects that are compared against the query. This can be achieved using either or both of the following techniques:

  - Aggressive Pruning: In this technique regions in the search space that are not likely to contain answers to the query are not accessed. An example of a method that uses this technique is a main memory indexing structure called the BBD-tree [10].

  - Early Stopping: These methods terminate when a maximum cost or an acceptable distance value is reached. An approximation which uses the M-tree [50] is presented in [215]. □

Approximate search methods are viewed as a trade off between speed and accuracy. [214]. These two criteria are assessed using three measurements; *efficiency*, *precision*, and *recall.* .

Efficiency: it is usually denoted as IE and defined as the cost ratio of the precise to the approximate query execution. This can be expressed as

$$IE = \frac{COST_{exact}(Q)}{COST_{approximate}(Q)} \tag{2-16}$$

Where $COST_{exact}$ and $COST_{approximate}$ represent the number of disk accesses needed to perform the exact and approximate search, respectively. Efficiency controls the speed at which the search query is performed. The accuracy of the search is controlled by precision and recall.

Precision: it is defined as the ratio of qualifying retrieved objects to the total number of objects retrieved by the approximate query. This can be formally specified as:

$$P = \frac{\left|S \cap S_{Approximate}\right|}{\left|S_{Approximate}\right|} \tag{2-17}$$

Where S is the response set of the exact search query and $S_{Approximate}$ is the response set of the approximate search query.

Recall: it is defined as the ratio of qualifying retrieved objects to the total number of qualifying objects, i.e.:

$$R = \frac{\left|S \cap S_{Approximate}\right|}{\left|S\right|} \tag{2-18}$$

2.9 Indexing Multimedia Databases

2.9.1 Introduction

In order to speed up the search process, similarity search algorithms build indices of the database in advance (offline). At query time, these indices are used to perform the similarity search more rapidly by avoiding a full scan of the database. Figure 2.9 illustrates this approach [38].

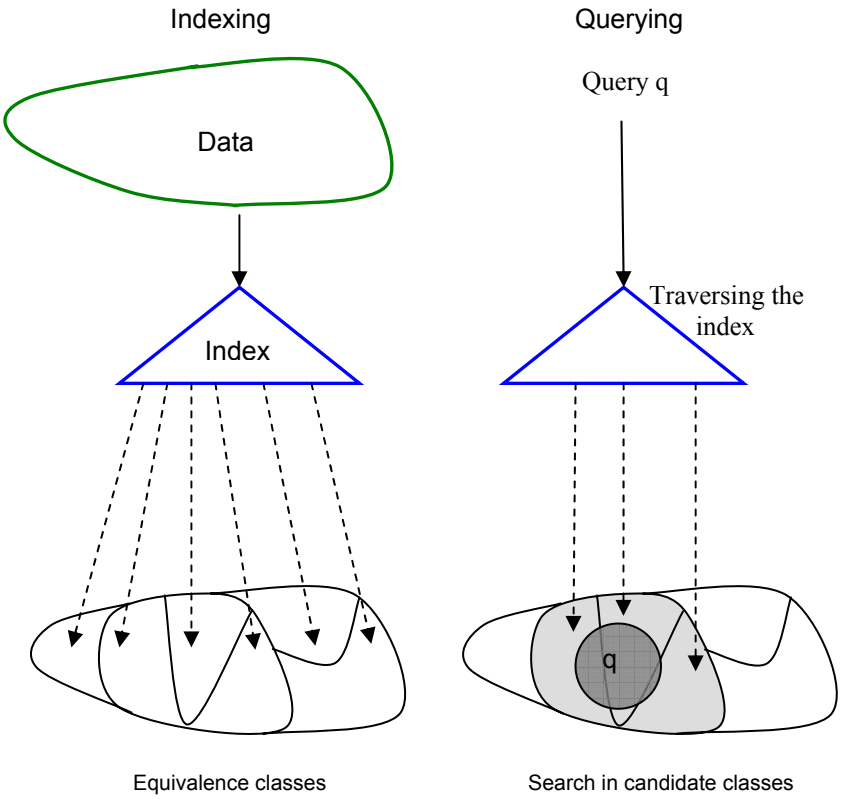


Fig. 2.9. The unified model for indexing and querying

In section 1.6 we presented an outline of existing indexing methods. In this section we present a more in-depth overview of these methods.

2.9.2 Dimensionality Curse

The term "dimensionality curse", which is also known by "Hughes effect" was first introduced in [15]. It refers to the fact that in order to estimate a function of several variables to a given accuracy, the number of data samples required grows exponentially as the number of dimensions grows linearly [109]. High-dimensional data spaces are inherently sparse, because available data are usually limited. This results in what is known as the *empty space phenomenon* [168].

2.9.3 Intrinsic Dimension

The *intrinsic dimension* (also named *intrinsic dimensionality*) of a vector can be defined as the minimal number of parameters or latent variables needed to describe it [109]. This concept, although seems simple, is more complicated to estimate in practice. In fact, intrinsic dimensionality does not refer to a single well-defined parameter, but rather to a family of parameters associated with a metric space [154].

2.9.4 Fractal Dimension

The *fractal dimension* is a powerful tool that can be used to describe the *data skew* of a dataset [149]. The term *fractal* means that the object does not need to exhibit exactly the same structure on all scales, but the same type of structures must appear [205]. The concept of fractal dimension has been used in range queries, as well as join queries [17], [157]. Fractal dimensions fit into a general paradigm of intrinsic dimensionality mentioned earlier in that fractal dimensions reflect the rate of growth of balls and boxes [39].

2.9.5 Indexing High-dimensional Spaces

What makes high dimensional spaces difficult to handle is that most of the effects resulting from the increase of dimensionality are unintuitive. While our perception of high-dimensional spaces is based on extending our image of low-dimensional spaces, this technique becomes deceptive when the dimension of the space becomes high, which is a part of the curse of dimensionality effect. In general, these effects can be classified into [23]:

1. Geometric effects concerning the shape of the hyper cubes or spheres
2. Effects concerning the shape and location of the index partitions
3. Effects concerning the database environment

2.9.6 Operations in High-dimensional Indexing Structures

The dynamic aspects of indexing structure are sustained using three main operations: delete, insert and update. The first two operations are the most important ones, since in a dynamic environment the databases are continuously submitted to adding some data objects and deleting others, so different indexing structures should be able to handle these operations easily and quickly. In some structures these operations are *local,* like in the hB-tree [116], for example, while in others they are *non-local*, an example of this is the R-tree [74].

2.9.7 High-dimensional Indexing Structures

Many structures have been proposed to manage high-dimensional data, which are not necessarily spatial. The principle is to use a suitable transformation to project these data onto lower dimensional spaces that can handle the data. Yet, even spatial indexing structures can fail to handle the data when the dimensionality increases. The reason for this is that by increasing space dimensionality more space is required to store a single data object. As a consequence, the index fanout (the number of children per node) is reduced considerably, resulting in an increase in disk accesses. In addition, the good properties of index structures no longer hold because of the excessive overlap, hence the discrimination power of the structure decreases [149].

In the following we present some of the most known indexing structures.

The R-tree: The R-tree [74] is a hierarchical structure that uses minimum bounding rectangles MBR, which are multidivisional intervals. This means that there is no smaller rectangle that can contain the point set in question. Overlapping is allowed, but it deteriorates the performance of the search.

The SS-tree: The SS-tree [206] uses spheres instead of rectangles as page regions. The center of the sphere is the centroid point of each dimension. The radius is chosen so that the sphere contains all the data objects.

The Space filling curves (SFC's): These are structures that are widely used in different fields of computer science, especially to linearize multidimensional data [117]. The most widely used SFC's are the Hilbert curve [88], the Z-curves [148], and the Gray code [64]. The SFC's embed a multidimensional space into a one-dimensional space while trying to preserve the original spatial proximity [11].

The Pyramid-tree: .The Pyramid-tree [19] uses a technique that is similar to that of the Hilbert curve, so it also maps a d-dimensional point into a 1-dimensional point, using a specific mapping called the *Pyramid-mapping.* The partitioning strategy is optimized for high-dimensional data, so the authors claim that their structure is the only structure not to be affected by the dimensionality curse.

2.10 Survey of Indexing Structures in Metric Spaces

2.10.1 Burkard-Keller Tree (BKT)

This tree [28] is considered as the first structure to present a general solution to similarity search in metric spaces. It belongs to ball partitioning methods (see 2.6.4). BKT is used with discrete distance functions and is built as follows; an arbitrary object $p \in U$ is selected as the root of the tree. For each distance $i \geq 0$ we define

subsets $U_i = \{u \in U, d(u, p) = i\}$. A child node of p is built for every nonempty subset U_i. Then all child nodes are partitioned recursively until there is only one object to process, or no more than b objects that we store in a bucket. The objects that are chosen as roots of the subtrees are called *pivots.* Figure 2.10 shows an example of BKT.

Range query is performed in the following way: The query $R(q, r)$ starts at the root node; it enters all the child nodes i that satisfy $|d(p, q) - r| \le i$. The algorithm continues recursively downwards until it reaches a leaf (or a bucket). This leaf or bucket is compared sequentially against the query.

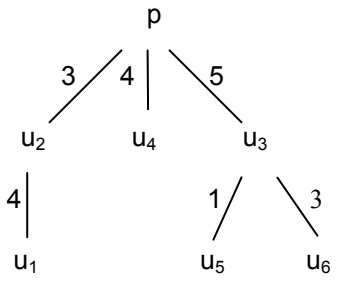


Fig. 2.10. BKT

The space complexity of BKT is $O(n)$, the construction complexity is $O(n \log n)$ (measured in terms of the number of distance computations required), and the search complexity is $O(n^{\alpha})$ (also measured by the number of distance computations needed), where α is real number that satisfies $0 < \alpha < 1$, and its value depends on the search radius and the structure of the tree [38].

2.10.2 Fixed Queries Tree (FQT)

This tree is another structure that utilizes ball partitioning. The *Fixed Queries Tree* was proposed in [13] as a modification of BKT tree. In FQT, a single pivot is used for all the nodes of a certain level. The advantage that this structure presents over BKT is that it decreases the number of distance computations, since, even if more than one subtree has to be accessed to evaluate the query, only one distance computation is computed between the query and the pivot of that level. Figure 2.11 shows an example of FQT built over the same data of Figure 2.10. The first level shows the distances between each of the points (u_2, p, u_3) and p. The second level (u_2, p, u_3) shows the distances between each of the points (u_2, u_5, u_6) and u_2.

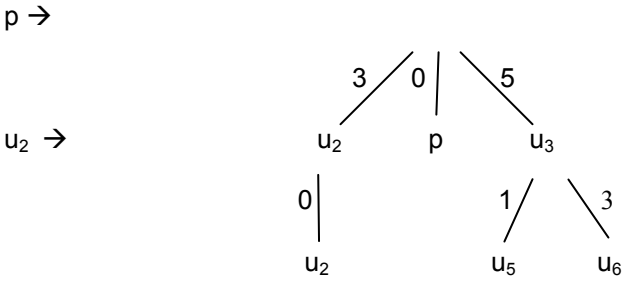


Fig. 2.11. FQT

There are a number of variants to this structure like the *Fixed-Height Fixed Queries Tree* (FHFQT), *Fixed Queries Array* (FQA), and others. All these structures handle discrete distance functions. FQT has the same complexity as that of BKT.

2.10.3 Vantage Point Tree (VPT)

The VPT [210] is designed to handle continuous distance functions. But it can also handle discrete distance functions. It is also based on ball partitioning.

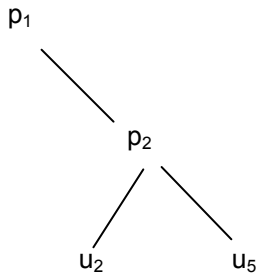


Fig. 2.12. VPT

This partitioning is based on the median of the set of all these distances, so those points whose distances from the vantage point are smaller than the median are inserted into the left subtree, and the other points are inserted into the right subtree. The process continues recursively until there is only one object in each subtree. This strategy of partitioning leads to a balanced binary tree. Figure 2.12 shows an example of VPT with two pivots. The first branch shows all the points in the datasets which satisfy $d(p_1,u) > m$, where m is the median of all the distances $d(p_1,u)$. This is repeated recursively where the points of each subree that are smaller than the median are inserted into the left subtree and the others to the right.

The construction complexity of VPT is $O(n \log n)$, and the query complexity is $O(\log n)$ but this is valid only for small values of r [210].

2.10.4 Bisector Tree (BST)

Unlike the structures we have discussed earlier, the BST [87] utilizes the generalized hyperplane partitioning. The BST is a binary tree that uses two pivots p_1 and p_2 to partition the data space. The objects that are closer to p_1 create the left subtree, and the objects that are closer to p_2 create the right subtree. Figure 2.13 shows an example of the first level of BST where (u_2, u_1, u_5) are closer to p_1 than to p_2, and (u_4, u_3, u_6) are closer to p_2 than to p_1.

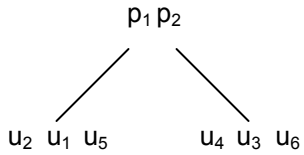


Fig. 2.13. BST

The construction complexity of BST is $O(n \log n)$, the query complexity is not reported.

2.10.5 Methods Exploiting Pre-computed Distances

The motivation behind these methods is that distance functions are usually expensive to calculate, so these methods suggest using pre-computed distances between data objects. Experiments show that this technique does enhance the search in terms of computational costs. Yet this technique has a drawback: the large storing space required overhead. However, this is not a serious problem with relatively small databases. In this section we present two of these methods; AESA, and LAESA.

AESA: The *Approximating and Eliminating Search Algorithm* (AESA) [194] uses a matrix of $n(n-1)/2$ distances between data objects. These distances have been pre-computed at construction time. In this structure all objects play the role of a pivot. At query time an object is chosen at random to be a pivot, and all data objects that satisfy the inequality $|d(q,p) - d(p,u)| > r$ are excluded. Then the algorithm chooses another object of the remaining objects to be a pivot and applies the above inequality,

and so on. Although this algorithm seems simple, the experiments show that it is competitive. However, AESA has a drawback, its high space complexity, which is quadratic.

Linear AESA (LAESA): The high space complexity of AESA has been overcome by a new structure; the linear AESA [138]. LAESA stores distances from data objects to only m pivots, so the distance matrix contains $m.n$ elements rather than $n(n-1)/2$ ones. The pivots are usually chosen in a way similar to that discussed in section 2.6.5.

2.10.6 Hybrid Methods

These methods combine pre-computed methods and partitioning methods, so these methods benefit from the advantages of both. In this section we present some of these methods.

Geometric Near-neighbor Access Tree (GNAT): The GNAT [27] is a structure that is based on the Voronoi Diagrams. In this structure a set of m centers $c_1, c_2, \ldots, c_m$ is chosen for each internal node and the set of data objects U is split into m subsets $U_1, U_2, \ldots, U_m$ (called Dirichlet domains) according to the shortest distance of the data objects to the m centers. Figure 2.14 shows an example of the first level of a GNAT with four centers.

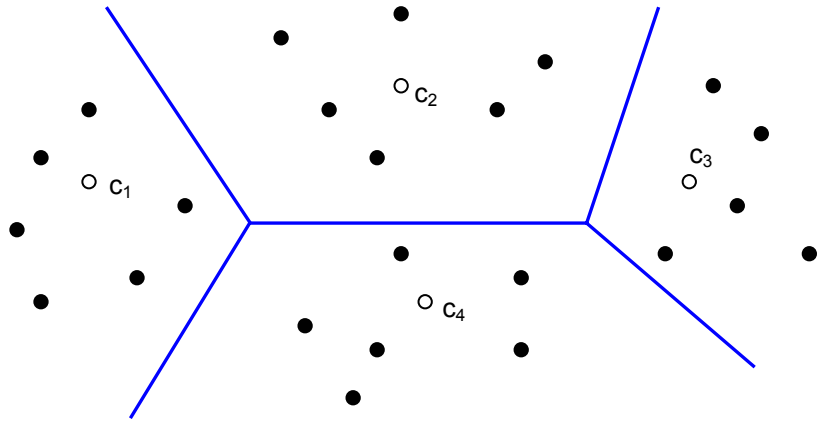


Fig. 2.14. GNAT

The algorithm stores, at indexing time, an $m \times m$ table that contains, in each cell, the minimum and maximum distances from each center to the objects in each subset U_m .At query time, the query is compared against some center, say c_i , and all classes whose centers do not intersect with $d(q,c) \pm r_i$, are discarded, then the algorithm chooses arbitrarily another center and continues in the same manner, until no classes can be discarded.

The construction complexity of GNAT is $O(nm \log_m n)$, the query complexity is not reported.

Spatial Approximation Tree (SAT) [145] The SAT is a search algorithm that approaches the query spatially, so this structure does not use pivots to partition the search space. To build this tree a data object c is chosen randomly to be the root of the tree. $N(c)$, the smallest set of all the neighbors of c, is built up as follows:

$$u \in N(c) \Leftrightarrow \forall u' \in N(c) / \{u\}; \quad d(u,c) < d(u,u') \tag{2-19}$$

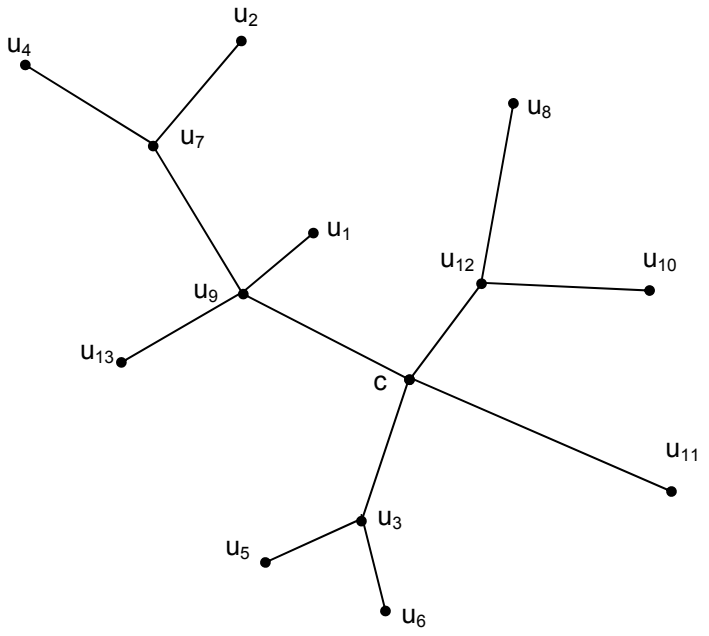


Fig. 2.15. SAT

In other words: all the data objects are sorted according to their increasing distance from c, the closest object is added to $N(c)$ if it is closer to c than it is closer to any object in $N(c)$, otherwise it will be assigned to that object which it is closer to. Then

each element of $N(c)$ is recursively the root of another subtree, together with the objects that they were assigned to it in the last step. Figure 2.15 shows an example of SAT with c as its root.

The construction complexity of SAT is $O\left(\frac{n \log n}{\log \log n}\right)$ and the query complexity is $O\left(n^{1-\theta\left(\frac{1}{\log \log n}\right)}\right)$.

Metric-tree (MT): The MT structure [50] aims at handling databases whose size changes dynamically. In these databases deletion and insertion operations are frequent. The MT, unlike other trees, is built in a bottom-up way. In the MT, all the objects are stored at the leaf node. This tree is similar to the GNAT, in that a set of representatives (pivots) are chosen at each internal node, and the objects closer to that representative are organized into a subtree, and the representative serves as a representative of the whole subtree. Each internal node is constrained by spheres and it contains a pointer to a child node, the covering radius of that sphere, and its distance from the associated parent node. At query time the query q is compared against all the representatives of the subtrees and enters those that can not be discarded using the covering radius.

The insertion of new objects is performed by choosing the subtree where the covering radius should expand minimally to contain that new object.

It is worth mentioning that the M-tree is the only structure that is optimized for secondary memory datasets, while the other structures mainly support main memory data sets .

The construction complexity of the MT is $O\left(n\left(m..m^2\right)log_m\, n\right)$. The query complexity is not reported.

Similarity Hashing (SH): This is a multi-tier hashing structure proposed by [71]. SH is a multi-level structure that consists of search-separable bucket sets on each level. This structure supports easy insertion and bounded search costs, because at most one bucket needs to be accessed at each level for range queries up to a pre-defined value of search radius. At the same time, the pivot-based strategy significantly reduces the number of distance computations. SH supports distributed and parallel implementations.

Hashing, which is also called *key-to-address transformation paradigms*, provides direct access to selected regions in the search space, which is something that tree-indexing structures can not offer.

2.11 Distributed and Parallel Systems

Although centralized search methods achieve speed-ups of orders of magnitude compared with sequential scanning, their performances deteriorates as the size of the database increases, which poses questions concerning the scalability of centralized methods.

Distributed and parallel search methods try to overcome this problem by utilizing dynamic heterogeneous systems that share resources such as computers, storage and data which are geographically distributed.

Parallel and distributed models share many important characteristics; shared memory models, interconnection networks and combinational circuits [188].

In parallel database systems the processors are tightly coupled in a single computer system [149]. Processors cooperate together to provide efficient processing of the similarity search. The user has no access to a specific processor of the system.

There are different parallel architectures, the processors may have access to a common memory, or they can communicate by message passing where the interconnection is achieved using high-speed links.

2.12 Unifying Model

Although all indexing structures seem to be different in nature, they are similar at heart, so in [38] the authors try to establish a unifying mathematical model that builds a common framework to all indexing structures. In this model the indexing algorithms aim at partitioning the search space U into subsets with relevant features. At query time the indexing structure should nominate a set of candidate subsets where the answers to the query are likely to be found, which corresponds to what is called the *internal complexity*, and then a post-processing scan on this set can eliminate all false alarms and return the true answers to the query, which corresponds to what is called the *external complexity*.

In this model, each partitioning scheme can be viewed as an equivalence relation (reflexivity, symmetry, and transitivity) and each class of this equivalence relation corresponds to a subset in the indexing structure.

2.13 Non-metric Similarity Search

The term non-metric means that the similarity function violates one, or all, the properties of a distance function [30].

Non-metric models are not widely explored yet, although they are attracting more and more attention lately as a more general model than the metric one. The problem with the non-metric model is that with the absence of the triangle inequality the search is mainly approximate [39].

In a very recent paper (to appear) [176] the authors propose a framework/map of the options that a domain expert can take when working with non-metric spaces:

- If the problem can be modeled in a metric space, there are plenty of solutions the choice of the best one of which depends on the data in question.
- If the problem uses a specific non-metric distance function it may have an available efficient index (BLAST [8], for example). These specific indices are usually more efficient than general ones.
- If the problem is modeled as black-box similarity (an algorithm returning a real-value output from a two-object input [119]) or if there is no specific index for this non-metric function then the options can be:
  - The problem can be mapped into another space or paradigm, although this may result in losing the discriminative power of the similarity function. This may also require a static database because this mapping requires pre-processing that can not be performed in dynamic environments.
  - One can also decide to use some of the few available general non-metric indexing structures or algorithms. However, this may result in slower query processing, besides the search algorithm returns approximate results only.

2.14 Summary

This chapter is a background chapter of the similarity search paradigm presented from a metric view point. We started by presenting some of the applications of this problems in different domains. Later we gave a formal presentation of the metric model and the distance metric as a measure of how two data objects are similar to each other. We also presented some of the widely-known metrics, mainly the Minkowsky family. Then we moved in the following section to presenting the different forms of the similarity search problem focusing on range queries, which will be dealt with often in later chapters. Next we presented how this problem has been addressed by researchers starting with the naïve sequential scanning approach with its high complexity then we showed how different techniques, such as the pivot technique and the compact partitioning, can be used to lower this complexity. We also

presented mapping methods which aim to reduce the complexity by processing the search in other spaces under certain conditions.

Although this dissertation handles exact solutions of this problem, approximate solutions are widely used in the literature, so they were presented briefly in this chapter together with some related issues to give the reader a thorough picture of the similarity search problem.

Indexing, a paradigm to speed up the similarity search, was also presented in this chapter in addition to some of the most widely used metric structures. We also gave a brief description of the concept of intrinsic and fractal dimensions as well as high-dimensional spaces.

Processing similarity search using distributed or parallel systems has been used more frequently lately. So we presented in this chapter some basic concepts of these approaches.

In the last part of this chapter we presented some efforts of establishing a unifying model to the search problem. Then in the last section we presented the non-metric model; a challenging topic in similarity search which tries to build a more general model than the metric one.

In the next chapters we see how time series data have been addressed through the metric model, but also through other data-specific models.

Although metric structures are not the topic of this dissertation, we think after examining the enormous number of structures that have been proposed to address the similarity search problem that research in this domain should focus now on finding novel algorithms for utilizing these structures rather than trying to introduce more new structures.

Chapter 3

Time Series Data Mining and Information Retrieval

This chapter concerns a data type, time series, which appears in a variety of applications in science, medicine and economics. The introduction of this chapter in Section 3.1 gives a formal definition of this data type in addition to some of its applications. In Section 3.2 we give a brief description of the main tasks in time series data mining. We also show the types of data transformation that are frequent with this data type and show the different techniques that are used to remove this transformation so that the results of the different tasks are more intuitive. Section 3.3 introduces time series retrieval and the types of queries this field of information retrieval deals with such as whole matching and subsequence matching. Section 3.4 starts with a presentation of a general framework to handle the similarity search problem in time series : the GEMINI algorithm, which most time series representation methods adopt to guarantee no false dismissals. The following subsection is a survey of the different representation methods, also called dimensionality reduction techniques, which are widely used in the time series community. The first technique we present is DFT which is probably the first dimensionality reduction technique in the literature. The next technique is DWT which uses wavelets; a powerful mathematical tool used to decompose signals. The third technique we present is SVD which is also used in many applications. This technique is very effective but its main drawback is its high complexity. PAA, a widely used technique, is presented in Subsection 3.4.4. In the following subsection PLA, an efficient but not indexable technique, is presented. In Subsection 3.4.6 APCA, an improvement of the PAA method is presented. CP, a technique that uses Chebyshev polynomials, is the last technique that we present in this section about representation methods. In the last subsection we show a comparison of the dimensionality reduction techniques we presented in this chapter.

In the second part of this chapter we revisit the concept of similarity distances used in the literature. So we present in Section 3.5 some metric distances like the Euclidean distance, and other non-metric distances like DTW and LCSS. We also show some of the techniques, like using lower bounds, which can be used to enhance these similarity measures that violate the triangle inequality. The last part of this section is devoted to state-of-the-art similarity measures and distance metrics in the field of time series. We conclude this chapter with a summary.

3.1 Introduction

A *time series*, also called *spatiotemporal trajectory*, is a collection of observations at intervals of time points. These observations are measurements of a particular phenomenon. If these time points are equally spaced, the time series is called *regular*, otherwise it is called *irregular*. Formally, an n-dimensional time series S is an ordered collection:

$$S = \{(t_1, v_1), (t_2, v_2), \ldots, (t_n, v_n)\} \tag{3-1}$$

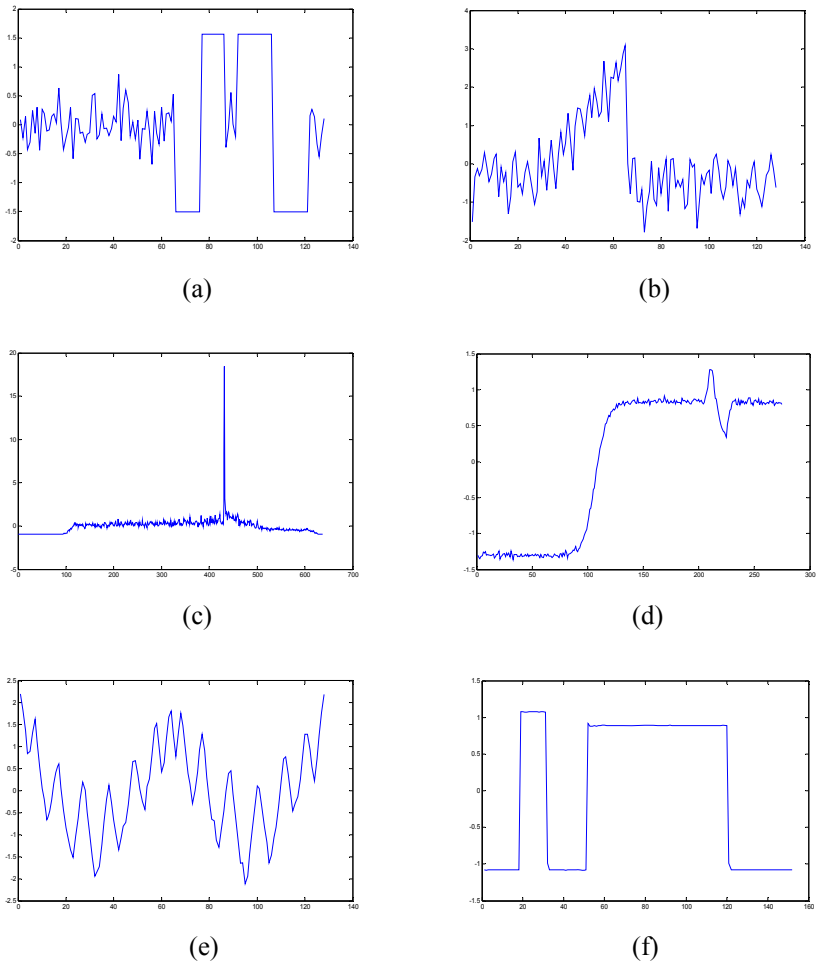


Fig. 3.1. Different examples of time series

where $t_1 < t_2 < ... < t_n$, and where v_i is the value of the observed phenomenon at time point t_i. The values v_i can be real numbers, or vectors, or even other data types (graphs, images, etc).

Sometimes a time series is represented by the values v_i that it takes.

Time series data appear in a broad variety of applications which vary from medicine, science and technology to business, finance and economics. Due to its numerous applications, this branch of computer science has witnessed increasing attention recently. Figure 3.1 shows some examples of time series.

Because of the several applications in which time series are involved, and the large size of time series databases, speed has always been the principal focus of all the methods and algorithms that handle this type of data.

In the literature, the terms *time series data mining* and *time series information retrieval* have usually been used interchangeably. In this dissertation we mainly use the term *time series information retrieval* to refer to the query by content problem over time series databases presented later in this chapter. This problem usually involves indexing methods. We reserve the term *time series data mining* to a wider class of problems in time series databases as indicated in Section 3.2. However, in many cases the distinction between the two terms is difficult since even some time series data mining tasks can involve indexing methods.

3.2 Time Series Data Mining

Data mining is a fundamental branch of computer science that witnessed a substantial progress in the last years. In [207] data mining is defined as "the process of discovering patterns in data. The process must be automatic or (more usually) semiautomatic. The patterns discovered must be meaningful in that they lead to some advantages. The data is invariably present in substantial quantities". Data mining is closely related to another branch of computer science which is *knowledge discovery*. However, the main focus of knowledge discovery is representing knowledge in a manner that is understandable to humans, while data mining focuses mainly on finding hidden information in large amounts of data.

In the following we present some of the most common data mining tasks [107], [140], [26]:

Description: In some cases, description of trends or patterns can explain them. High quality description can often be obtained by using exploratory data analysis (EDA), which is a graphical method that helps explore the data in search of patterns and trends.

Data Preparation or Data Pre-processing: Most of the raw data are unprepared, noisy, or incomplete. All this makes data mining systems fail to process these data properly that a preparation phase seems inevitable before handling these data. This phase may include different processes such as data cleansing, normalizing, handling outliers, completion of missing values, and deciding which attributes to keep and which ones to discard.

Classification: In classification we have categorical variables which represent classes, and the task is to assign class labels to the dataset according to a model learned from a learning phase on a training data where the classes are known (supervised learning). Then the algorithm looks at new data and tries to classify these data based on the model acquired during the training phase.

Estimation: Estimation (known also as regression, mainly in the statistics community) is similar to classification, except that in estimation the target variables are numeric. The model is built using target variables and predictors, and then for new observations the value of the target variable is estimated based on the values of the predictors. This is equivalent to what is known as *interpolation* in the numerical analysis community.

Prediction: This task also includes a kind of estimation, except that it concerns values that are beyond the range of already observed data. This is equivalent to *extrapolation* in numerical analysis.

Query by Content: In this task the algorithm searches for all the objects in the data base and returns all those which are similar to a given pattern (see Chapter 2).

Clustering: This is a process in which the algorithm groups the data objects into classes of similar objects. Clustering is different from classification in that in clustering we do not have target variables. In other words, clustering is a process of partitioning the data space into groups of similar objects.

Association Rules: This is the task of finding relationships between two or more attributes. These rules have the form: "*if… then…* ", together with a measure of confidence associated with these rules.

3.2.1 Data Transformation

Time series data mining addresses the problems that are particular to this data type. Distortion is the most frequent problem in time series databases. This problem can have different forms: noise (Figure 3.1, a), amplitude scaling (Figure 3.1, b) amplitude shifting (Figure 3.1, c), time scaling (Figure 3.1, d) , and outliers (Figure 3.1, e).

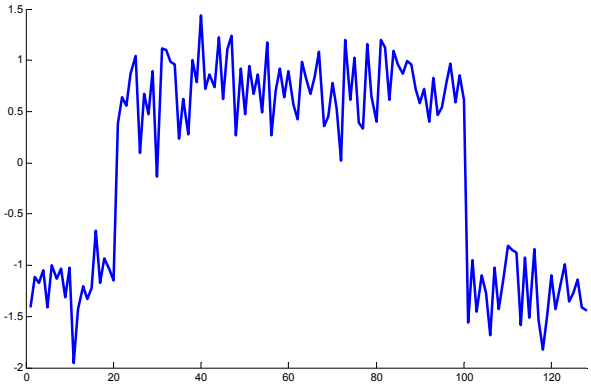


Fig. 3.2 (a). Noise

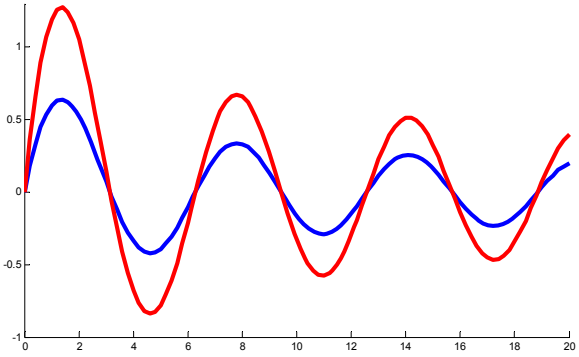


Fig. 3.2 (b). Amplitude scaling

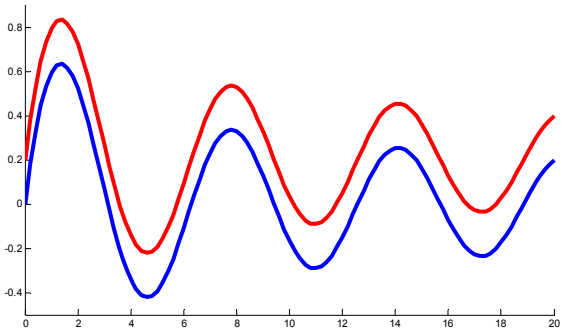


Fig. 3.2 (c). Amplitude shifting

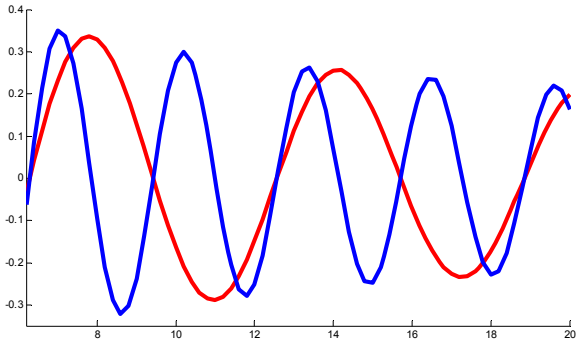


Fig. 3.2 (d). Time scaling

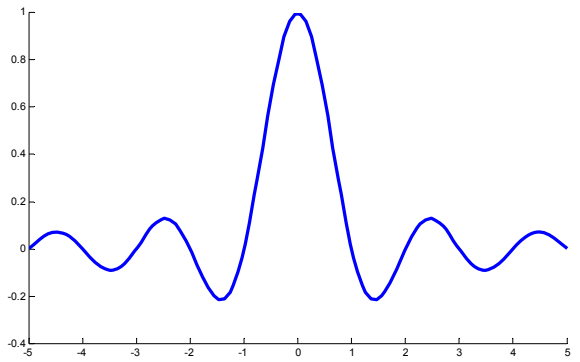


Fig. 3.2 (e). Outlier

Applying time series data mining tasks on raw data may result in unintuitive results. For instance, the Euclidean distance is widely used in time series data mining and information retrieval, but it is sensitive to noise and to shifts on the time axis [126]. That is why pre-processing of time series is necessary before applying different time series data mining algorithms.

There are different techniques that can be used to eliminate distortions, depending on their nature. Noise is usually removed by smoothing the data which is achieved by taking the average of several successive data points. Amplitude shifting and amplitude scaling are removed by normalizing the data. The most common type of normalization is the *z-score standardization* in which the mean of the normalized values is obtained by subtracting the mean value from the raw (unprocessed) value, and dividing the outcome by the standard deviation, i.e.

$$v_{norm} = \frac{v_{raw} - mean(v)}{std(v)} \tag{3-2}$$

where v_{raw} is the observed value (see 3.1), v_{norm} is the z-score standardization

Outliers, on the other hand, are not easy to identify. The z-score standardisation can be used for this purpose, as a value can be considered an outlier if it is farther from the mean than 3 standard deviations; i.e. its z-score standardisation is less than -3 or greater than +3.

The problem with using the mean and the standard deviation in the z-score standardization formula is that both these statistical measures are sensitive to outliers. Therefore, some researchers use another statistical method to detect the outliers which is the *interquartile range* (IQR) [107], which is defined as:

$$IQR = Q_3 - Q_1 \tag{3-3}$$

where Q_1 is the *first quartile*, i.e. the 25th percentile , and Q_3 is the *third quartile*, i.e. the 75th percentile.

Using the above concept, a data value is considered an outlier if it is located 1.5(IQR) or more below Q_1 or if it as located 1.5(IQR) or more above Q_3.

3.3 Time Series Information Retrieval

As indicated earlier, in this problem a query time series is given and the search algorithms returns all the time series that are similar to that given query. This problem comes in two "flavors": whole matching queries and subsequence matching queries [44].

Whole matching:

Given a time series database D of m time series of length n, a query time series Q (not necessarily from D) of length n too, a real number $r \geq 0$ which represents a threshold, and a distance function d defined on D. The whole matching query is the set of all the time series $S \in D$ which satisfy:

$$d(S, Q) \leq r \tag{3-4}$$

Notice that in this problem the time series in the databases, in addition to the query time series, all have the same length n.

Subsequence matching:

Given a time series database D of m time series of arbitrary length, a query time series Q (not necessarily from D) of length $l \le l_{\min}$, where $l_{\min}$ is the length of the shortest time series in D, a real number $r \ge 0$ which represents a threshold, and a distance function d defined on D. The subsequence matching query reports all the time series $S \in D$ together with the offsets within these time series that satisfy:

$$d\left(S_{[i:(i+l-1)]}, Q\right) \le r \qquad (3\text{-}5)$$

3.4 Dimensionality Reduction Techniques

Data structures such as the R-tree and its variants (see 2.9.7) can be used to handle the problem of time series information retrieval. Nevertheless, time series are high dimensional data [75]. This high dimensionality can cause these indexing structures to fail in handling these data. On the other hand, time series are highly correlated data, so dimensionality reduction techniques, also known as *representation methods*, try to benefit from this fact to find a faster solution to the similarity search problem in time series databases. Dimensionality reduction techniques aim 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.

Tables 3.1. The GEMINI algorithm for time series range queries

Algorithm: range_query(Q,r)

1. Transform the time series in the database DB from the original n-dimensional space into a lower dimensional space of N dimensions

2. Define a lower bounding distance on the reduced space: $d^N(S_i, S_j) \le d^n(S_i, S_j) \quad \forall S_i, S_j \in DB$

3. Eliminate all the time series for which we have $d^N(Q,S) > r \rightarrow$ obtain a candidate response set

4. Apply d^n to the candidate response set and eliminate all the time series that are farther than r from Q to get the true response set.

The GEMINI Framework: In [65] the authors presented a generic approach for indexing time series. Later GIMINI was extended to other data types. GEMINI reduces the dimensionality of the time series by converting them from a point in an *n*-dimensional space into a point in an *N*-dimensional space, where $N<<n$. A similarity distance is defined on the reduced space, which is lower bounding to the original similarity distance, thus the similarity search returns no false dismissals in this case. A post-processing sequential scan on the candidate response set is performed to filter out all false alarms and return the true response set. Table 3.1 illustrates the GEMINI algorithm.

3.4.1 Discrete Fourier Transform (DFT)

This is probably one of the first dimensionality reduction techniques known in the literature. *Discrete Fourier Transform* (DFT) is one of the specific forms of Fourier analysis. It transforms one function of the original form (which is often a function in the time domain) into another, which is called the frequency domain [143].This transform as a method of indexing time series was presented in [4], [5]. The basic idea of DFT is that any complex time series or signal can be expressed in terms of sine/cosine waves. Each time series can be represented using complex numbers called the *Fourier Coefficients*. So a time series of 256 dimensions, for instance, can be represented by 128 complex Fourier coefficients. However, the first coefficients are the most significant and the most representative ones, so the other Fourier coefficients can be truncated without much loss of information. This makes DFT an efficient dimensionality reduction technique with a good compression ratio that a compression ratio of 1:16 can be achieved [91], where the compression ratio represents the ratio of the dimension of the original space to the dimension of the reduced space.

The original algorithm has a high complexity $O(n^2)$. But a well-known efficient algorithm named *Fast Fourier Transform* (FFT) has a lower complexity of $O(n \log n)$ [171].

The main property of Parseval's theorem [147] is that the Euclidean distance between two time series in the time domain is the same as their Euclidean distance in the frequency domain. This property is the basis of using DFT as an indexing method in time series information retrieval.

The distance between two time series X , Y proposed by the authors of [4] is:

$$d(X,Y) = \sqrt{\sum_{t=0}^{n-1} \left| x_t - y_t \right|^2} = \sqrt{E(X-Y)} \qquad (3\text{-}6)$$

This is the distance in the original space. By using few DFT coefficients to compute the Euclidean distance we get a distance that is lower bounding to the Euclidean distance presented in (3-6)

Notice that DFT requires that the two time series have the same length.

It is worth mentioning that other variations of DFT take the best coefficients instead of the first few coefficients. Another variation called DCT uses only the cosine waves [91].

To illustrate DFT as an indexing method, let us see how it is applied to the time series presented in Figure 3.3 (This is the same time series shown in Figure 3.1 (e))

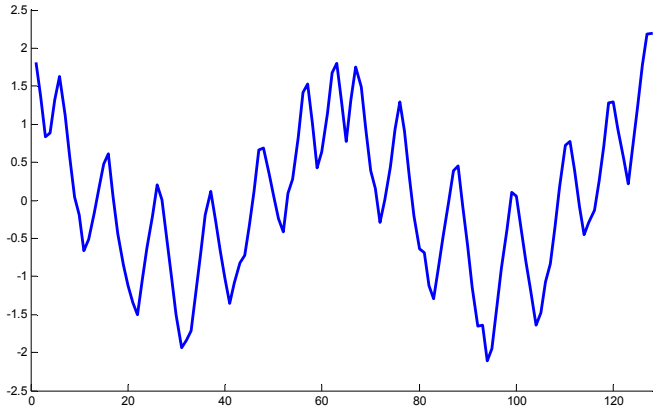


Fig. 3.3 (a). The original time series on which DFT is applied

This is a 128-dimensional time series. By taking the first 8 coefficients we get the following waves (Figure 3.3 (b)):

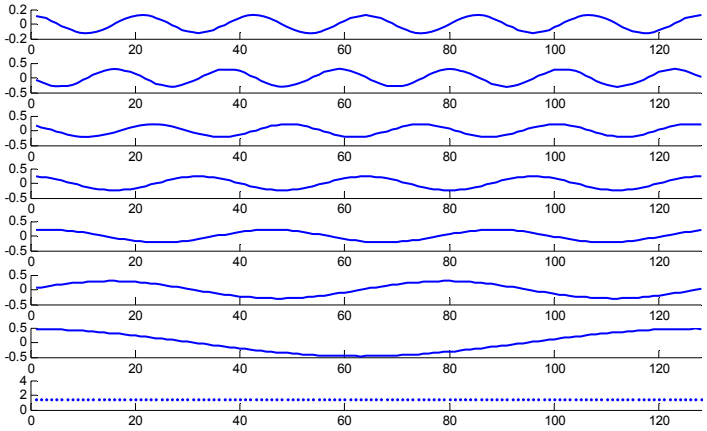


Fig. 3.3 (b). The first 8 coefficients of DFT

Using these waves, the original time series shown in Figure 3.3 (a) can be recovered to get the lower-dimensional representation shown in Figure 3-3 (c).

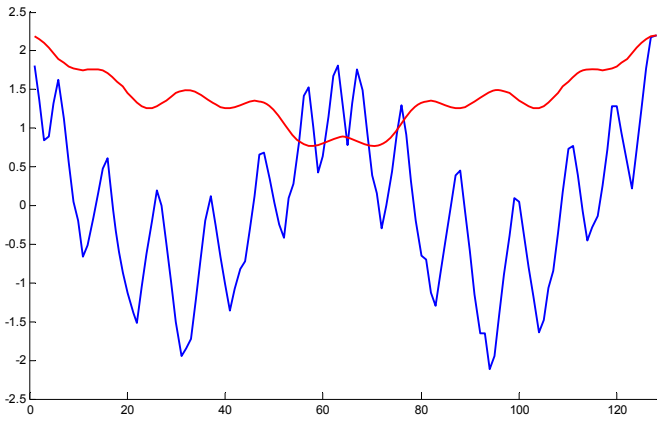


Fig. 3.3 (c). The lower-dimensional time series using 8 coefficients of DFT

3.4.2 Discrete Wavelet Transform (DWT)

Wavelets are a mathematical tool for hierarchically decomposing functions. Regardless of whether the function of interest is an image, a curve, or a surface, wavelets offer an elegant technique for representing the levels of details present [183]. Wavelets have successfully been used in many fields of computer science such as image compression [55], image querying [84], and many others.

DWT has also been used in time series information retrieval as a dimensionality reduction technique [35], [155], [208]. The advantage that DWT has over DFT in indexing time series data is that DWT is a multi-resolution representation method and it can represent local information in addition to global information.

Haar wavelets are the simplest form of wavelet. Haar wavelet transform is a series of averaging and differentiating operations. To get an idea of how 1-dimensional Haar wavelets work, let us consider the following 4-dimensional time series: $S = [8,4,3,5]$. By taking the average of each two successive values we get the following 2-dimenisonal time series: $S' = [6,4]$.

To recover the original 4-dimensional time series we need to use the DWT *Coefficients*, which, in this case, are: $2 = \frac{8-4}{2}$ and $-1 = \frac{3-5}{2}$.

Recursively repeating this process we get the full decomposition of S as shown in the table:

Resolution	Averages	*DTW* Coefficients
4	[8,4,3,5]	
2	[6,4]	[2,-1]
1	[5]	[1]

So the wavelet transform of S is $[5,1,2,-1]$.

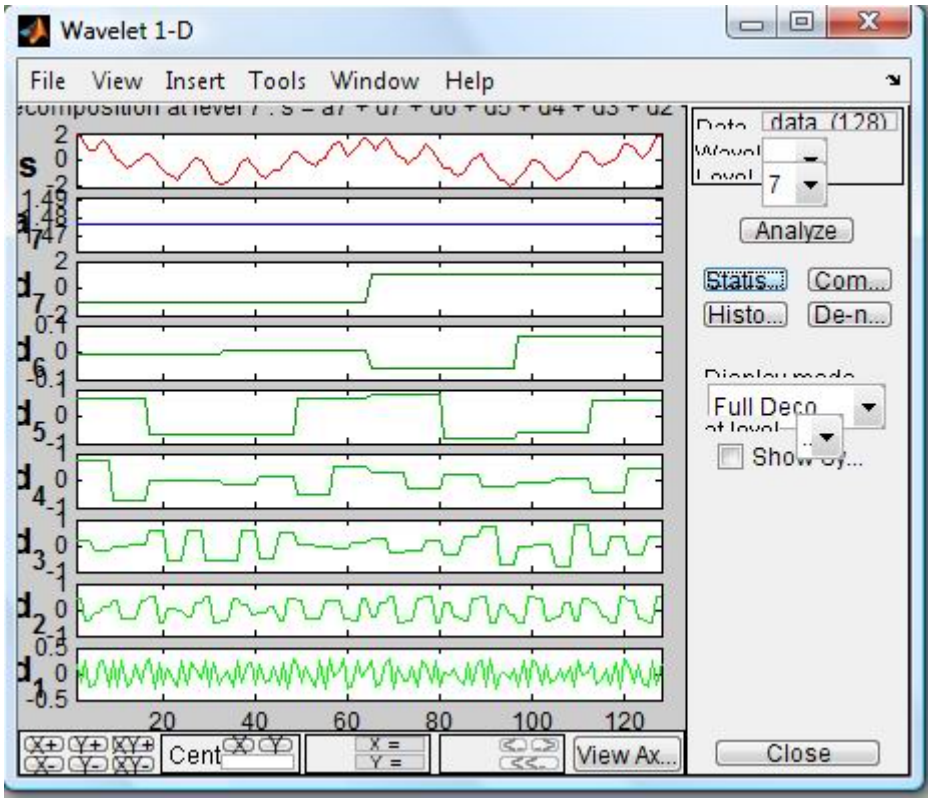
The basic idea of DWT is similar to that of DFT in that a time series can uniquely be represented by a wavelet transform, but by keeping only the first N coefficients we can reduce the dimensionality and keep much of the information that is in the original time series. For instance, Figure 3.4 shows the DWT decomposition at level 7 of the time series show in Figure 3.1 (e). To easily compare the representation of DWT with that of DFT, we show separately one of these levels using the same scale that was used in Figure 3.3.

A lower bounding distance to the Euclidean distance was presented in [35] and it was proven that this lower bound guarantees no false alarms.

It is also proven in [35] that the complexity of Haar transform is $O(n)$, which is lower than that of DFT.

It is important to mention that DWT requires that the length of the time series be a power of 2.

There exist controversies on whether other wavelets are better for indexing. In [155] the authors claim that Daubechies wavelets outperform Haar wavelets, while in [35] the authors claim the contrary.



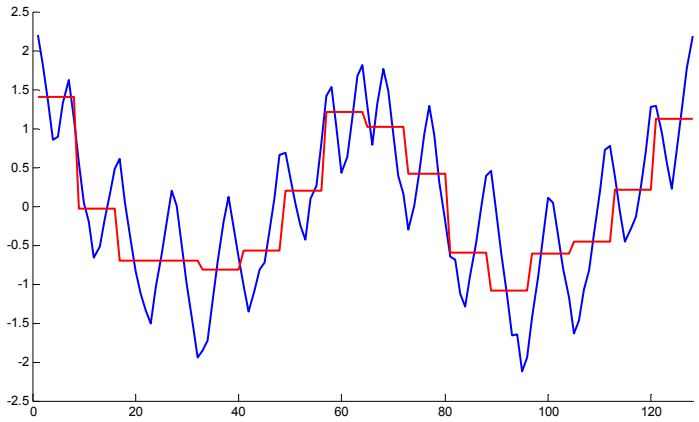


Fig. 3.4. The lower-dimensional representation of the time series using DWT

3.4.3 Singular Value Decomposition (SVD)

Singular value decomposition (SVD) has been widely used in different fields of computer science such as text retrieval, where it is called *Latent Semantic Indexing* [61] , pattern recognition [60] , face recognition [189] , and image databases [160], [146]. It has also been proposed as a dimensionality reduction technique in time series information retrieval in [104].

At heart, SVD is similar to DFT and DWT in that it represents the time series as a linear combination of eigenvalues but keeps only the first N coefficients as the lower-dimensional representation of the original time series.

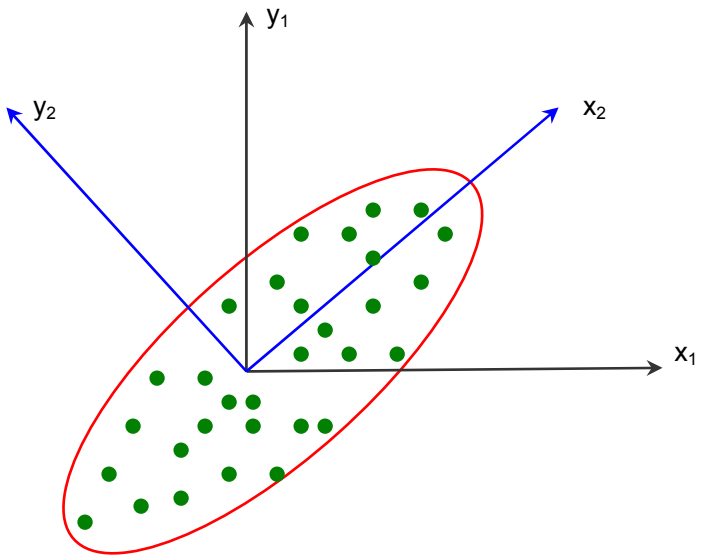


Fig. 3.5. SVD in a 2-dimensional space

The intuition behind SVD is that a set of m points will group on few dimensions only. Each group represents a "principal component". This component has a high discriminatory power and is orthogonal to other groups. SVD finds these dimensions and projects the points onto these dimensions. Figure 3.5 explains this idea in a 2-dimensional space. By rotating the axis ox_1y_1 to ox_2y_2 the best direction for projection is ox_2 the second best is oy_2. In other words, ox_2 is the direction of maximum variance, oy_1 is the direction of maximum variance orthogonal to oy_2 .

Mathematically, SVD can be expressed as follows: Given a time series databases which contains m time series whose dimension is n, the SVD composition of this database is a matrix X :

$$X = U\Sigma V^T \tag{3-7}$$

where U, V are orthogonal. Σ diagonal, and Σ contains the eigenvalues of $X^T X$.

The first few eigenvalues contain most of the variance of the time series, so the idea is to keep as many eigenvectors as space permits. These retained terms are called the *k principals components* [104].

SVD outperforms both DFT and DWT, but its complexity is very high $O(mn^2)$. Another disadvantage of SVD is that the transform requires that all the data in database be available, which means that the database should not be updated too frequently [44].

3.4.4 The Piecewise Aggregate Approximation (PAA)

This method was proposed in [92] and [212], independently. Its basis is simple and straightforward, yet this method has been successfully used as a competitive method. In fact, two other dimensionality reduction techniques have been derived from PAA: *Adaptive Piecewise Constant Approximation* (APCA) [93] and the *Symbolic Aggregate Approximation* (SAX) [111].

PAA reduces the dimensionality of a time series *S* from *n* in the original space to *N* in the reduced space. This is achieved 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. Figure 3.6 shows an example of PAA applied to a time series to reduce its dimensionality from a 12-dimensional space into a 3-dimensional space

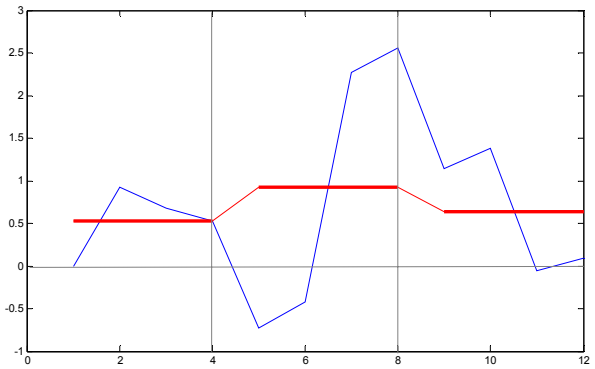


Fig. 3.6. PAA from a 12-dimensional space into a 3-dimensional space

The similarity distance used in the reduced space is:

$$d^N(X,Y) = \sqrt{\frac{n}{N}}\sqrt{\sum_{i=1}^{N}(\bar{x}-\bar{y})^2} \tag{3-8}$$

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 l is n/N , so l is the length of each segment.

It is proven in [92] and [212] that the above similarity distance is a 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 GEMINI, all time series that satisfy:

$$d^N(Q,S) > r \tag{3-9}$$

Can not be answers to the query and should be excluded.

3.4.5 The Piecewise Linear Approximation (PLA)

The *Piecewise Linear Approximation* (PLA) was presented as a method to index time series in [172]. However, this proposed method did not guarantee no false dismissals. In [141] the authors proposed a novel indexing structure based on PLA called the *L-index*. In this method the time series is transformed into Δ-SEALS (Δ-*bounded Sequence of Approximated Liner Segments*). The basic idea of the Δ-SEALS is to approximate the time series by a sequence of k linear segments. Each line segment is the longest possible linear segment whose accumulated error does not exceed a given deviation bound Δ, where the error is defined by the least square method. Figure 3.7 shows how the Δ-SEALS are built.

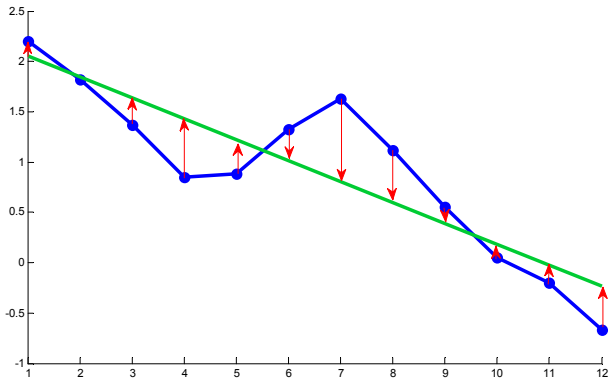


Fig. 3.7. The Δ-SEALS

The authors present two distance measures; the first is called the *Optimistic Bound Distance* (OBD). A bounding value of the error deviation called the *worst error deviation* (wed) is used to guarantee that the proposed distance underestimates the distance in the original space, thus the method with this distance produces no false dismissals.

The authors also define another distance measure called the *Pessimistic Bound Distance* (PBD) .This method overestimates the distance defined in the original space, so this distance produces no false alarms.

PLA is a competitive method, but one of its cons is that there was no proposed method to index it. This reduces its efficiency when used over very large databases.

In [45] the authors present an *Indexable Piecewise Linear Approximatio*n method (IPLA). This method uses a new distance function with PLA. The authors prove that this distance is lower bounding to the Euclidean distance on the original space. They also present a scheme to index their proposed distance. The experiments conducted by the authors show that indexable PLA outperforms other state-of-the-art methods

3.4.6 The Adaptive Piecewise Constant Approximation (APCA)

This method was presented in [93] as an extension to PAA. While PAA uses frames of equal lengths, APCA relaxes this condition and represents each time series in the database by a set of constant value segments of varying lengths such that their individual reconstruction errors are minimal. The intuition behind this idea is that different regions of the time series contain different amounts of information. So while regions of high activity contain high fluctuations, other regions of low activity show a flat behavior, so a representation method with high fidelity should reflect this difference in behavior.

Given a time series $S = \{v_1, v_2, ..., v_n\}$. Its APCA representation is:

$$C = \{\langle cv_1, cr_1\rangle, ..., \langle cv_N, cr_N\rangle\} \tag{3-10}$$

where $cr_0 = 0$ and where cv_i is the mean value of the data points in the ith segment, i.e. $cv_i = mean\ (C_{cr_{i-1}+1}, ..., C_{cr_i})$, cr_i is the right end point of the ith segment, and N is the number of segments. We do not represent the length of the segments but the locations of their right endpoints are recorded instead for indexing reasons.

The authors of [93] propose two distance measures to be used in the reduced space; the first d_{AE} is non-lower bounding of the Euclidean distance, but a very tight approximation of the Euclidean distance and can support fast approximate search. The second distance d_{LB} is lower bounding of the Euclidean distance and can support exact search.

Given a query time series Q and a time series S whose APCA representation is C. The approximate Euclidean distance d_{AE} is defined as follows:

$$d_{AE}(Q,C) = \sqrt{\sum_{i=1}^{N} \sum_{k=1}^{cr_i - cr_{i-1}} (q_{k+cr_{i-1}} - cv_i)^2} \tag{3-11}$$

This distance measure can be calculated in $O(n)$.

It is proven in [93] that this distance measure does not satisfy the triangle inequality. This means that it may not be a lower bounding of the Euclidean distance

The lower bounding distance measure d_{LB} is defined using a special version of APCA. This version is obtained by projecting the end points of C onto Q then finding the mean value of the sections of Q that lie within the projected intervals. The APCA representation of Q obtained using this version is denoted by Q'. d_{LB} is defined as follows:

$$d_{LB}(Q',C) = \sqrt{\sum_{i=1}^{N} (cr_i - cr_{i-1})(qv_i - cv_i)^2} \tag{3-12}$$

It is proven that this distance lower bounds the Euclidean distance.

3.4.7 Chebyshev Polynomials (CP)

This method was presented in [31]. It uses Chebyshev polynomials to represent the time series since it is shown in [125] that Chebyshev approximation is almost identical to the optimal minimax polynomial, it is also easy to compute.

The Chebyshev polynomial is a polynomial in t of degree m and defined as follows:

$$P_m(t) = \cos(m.\cos^{-1}(t)), \qquad t \in [-1,1] \tag{3-13}$$

This can be rewritten with the following recurrence relation:

$$P_m(t) = 2t.P_{m-1}(t) - P_{m-2}(t) \qquad \forall m \geq 2 \tag{3-14}$$

where $P_0(t) = 1, P_1(t) = t$

Chebyshev polynomials are orthogonal so they can be used as a base to approximate any function.

Interval functions are functions whose domain is an interval (which is $[-1,1]$ in this case). The functions may be continuous or not, but they should be defined everywhere over the interval.

Time series is discrete, so it should be converted into an interval function. To do so, the time series $S = \{v_1, v_2, ..., v_n\}$ should be rewritten in a functional form as follows:

$$S(t) = \begin{cases} v_i & \text{if} \quad t = t_i \\ \text{undefined} & \text{otherwise} \end{cases} \tag{3-15}$$

Then the $[-1,1]$ interval is divided into n disjoint subintervals as follows:

$$I_i = \begin{cases} \left[-1, \dfrac{t_1 + t_2}{2}\right[& \text{if} \quad i = 1 \\ \left[\dfrac{t_{i-1} + t_i}{2}, \dfrac{t_i + t_{i+1}}{2}\right[& \text{if} \quad 2 \le i \le n-1 \\ \left[\dfrac{t_{n-1} + t_n}{2}, 1\right] & \text{if} \quad i = n \end{cases} \tag{3-16}$$

The following step function can be chosen as an interval function :

$$g(t) = v_i \qquad \text{if} \quad t \in I_i \qquad \text{for} \quad 1 \le i \le n \tag{3-17}$$

However, this function does not guarantee the lower bounding lemma. To satisfy the lower bounding lemma the interval function should be:

$$f(t) = \frac{g(t)}{\sqrt{w(t)|I_i|}} \qquad \text{if } t \in I_i \qquad \text{for} \quad 1 \le i \le n \tag{3-18}$$

$w(t)$ is the *Chebyshev weight function* and is defined by:

$$w(t) = \frac{1}{\sqrt{1-t^2}} \tag{3-19}$$

After that, the Chebyshev coefficients can be computed by the following formulas:

$$c_0 = \frac{1}{n}\sum_{j=1}^{n} f(t_j)P_0(t_j) = \frac{1}{n}\sum_{j=1}^{n} f(t_j) \tag{3-20}$$

$$c_i = \frac{2}{n}\sum_{j=1}^{n} f(t_j)P_i(t_j) \tag{3-21}$$

It is shown in [31] that the computational complexity of Chebyshev polynomial is $O(n)$.

Given two time series S_1, S_2 of length n, let $\vec{C}_1, \vec{C}_2$ be their corresponding vectors of Chebyshev coefficients, respectively. i.e. $\vec{C}_1^T = [a_0, a_1, ..., a_m]$, $\vec{C}_1^T = [b_0, b_1, ..., b_m]$, where T denotes the vector's transpose. The distance function between the two vectors of Chebyshev coefficients is defined as follows:

$$d_{chy}(\vec{C}_1, \vec{C}_2) = \sqrt{\frac{\pi}{2}\sum_{i=0}^{m}(a_i - b_i)^2} \tag{3-22}$$

The authors of [31] show that this distance is metric and that it is lower bounding to the Euclidean distance in the original space.

3.4.8 Comparison of the Different Representation Techniques

With all these dimensionality reduction techniques that we covered in this chapter and others that we did not cover a question arises; which dimensionality reduction technique is the best? Almost every paper with a new dimensionality reduction technique claims that this new technique is the most effective and the most efficient. In [155], for instance the authors claim that wavelets outperform DFT, while the authors of [89] claim that the filtering performance of DFT is superior to that of DWT, and in [208] the authors claim that DFT and DWT give comparable results. There are similar contradictions concerning the performance of other methods.

The datasets used in the experiments seem to play a role in favoring one method over another. Some other aspects should also be taken into consideration when comparing dimensionality reduction techniques; SVD, for instance, is competitive, but it is very costly computationally. DWT is applied to time series whose length is of power of 2, etc. The similarity measure used in the experiments may also influence the results.

Tables 3.2. Comparison of different representation methods

The Method	Computational Complexity	Space Complexity	Index-ablility	Length Limitation	Supporting Weighted L_p Distances
DFT	$O(n(\log(n))$	$O(n)$	Yes	No	No
DWT	$O(n)$	$O(n)$	Yes	Yes (power of 2)	No
SVD	$O(Nn^2)$	$O(Nn)$	Yes	No	No
PAA	$O(n)$	$O(n)$	Yes	No	Yes
PLA	$O(n)$	$O(n)$	No*	No	Yes
APCA	$O(n)$	$O(n)$	Yes	No	Yes
CP	$O(n)$	$O(n)$	Yes	No	Yes

* but IPLA is indexable

In [57] the authors conducted extensive experiments implementing different representation methods and using different similarity measures. The authors reach at a conclusion that the pruning efficiency of different representation methods is almost the same. At any rate, we think that the results of such experiments should not be taken for granted if we take into account that many of such experimental comparison papers are written by authors who have proposed a representation method or a similarity measure.

However, we are tried to make a comparison of the different methods based strictly on the theoretical features of the representation methods. But it is important to state that the comparison based on theoretical features is not "better" than experimental comparison, because in some cases the theoretical feature seems to represent the worst case or the best case evaluation. In Table 3.2 we present a comparison of the representation methods we presented in this chapter [44].

3.5 Similarity Distances in Time Series Information Retrieval

Research in time series information retrieval has focussed on two aspects: dimensionality reduction methods, and similarity distances. There have been several proposed distances to compute the distance between time series, some are distance metrics while the others are similarity measures that represent a weaker form of similarity. In the following we present some of these distances.

3.5.1 The Euclidean Distance

This distance is a member of the L_P family (see section 2.4). It is the first distance measure used in time series information retrieval [4]. The Euclidean distance is effective [95], [161] and it has been widely used [36], [92], [93].

However, the Euclidean distance has a few inconveniences: it is sensitive to noise and to shifts on the time axis and thus lacks elasticity [47], [86], [103]. It is also applied to series of identical lengths only [126]. Figure 3.8 shows an example of the Euclidean distance between two time series.

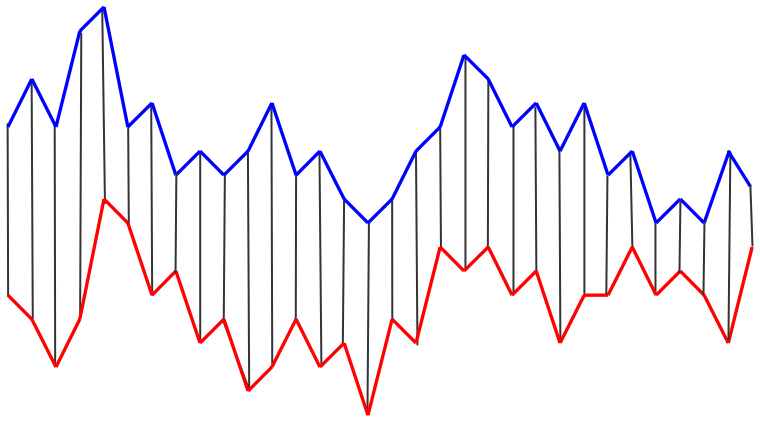


Fig. 3.8. The Euclidean distance between two time series

One of the variations of the Euclidean distance is the *Weighted Euclidean Distance*, which is defined between the two time series $S = \{s_1, s_2, ..., s_n\}$, $R = \{r_1, r_2, ..., r_n\}$ as follows:

$$d(S, R, W) = \sqrt{\sum_{i=1}^{n} w_i (s_i - r_i)^2} \tag{3-23}$$

where W is the weight vector.

The intuition behind this distance is that some parts of the time series may have more importance than other parts. Figure 3.9 illustrates the weighted distance.

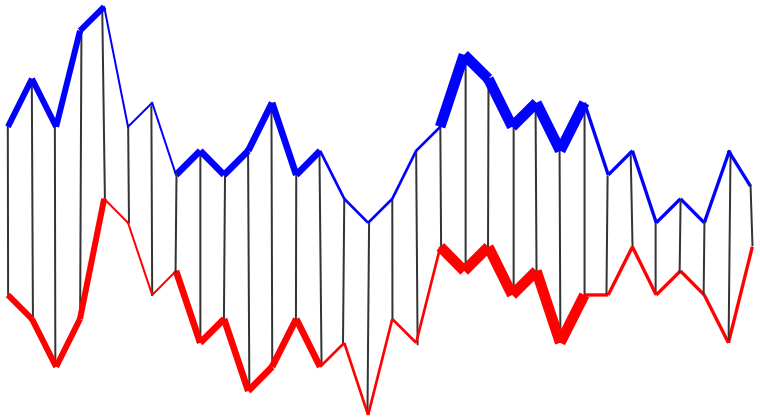


Fig. 3.9. The weighted distance

3.5.2 Dynamic Time Warping (DTW)

The *dynamic time warping* has been developed by the speech recognition community [193], [162], and later was used with time series [22]. DTW is an algorithm to find the optimal path through a matrix of points representing possible time alignments between the signals. The optimal alignment can be efficiently calculated via dynamic programming [73].

The dynamic time warping between the two time series $S = \{s_1, s_2, ..., s_n\}$, $R = \{r_1, r_2, ..., r_m\}$ is defined as follows:

$$DTW(i,j) = d(i,j) + min\begin{cases} DTW(i,j-1) \\ DTW(i-1,j) \\ DTW(i-1,j-1) \end{cases} \quad (3\text{-}24)$$

where $1 \le i \le n, 1 \le j \le m$. Notice that the two time series need not have the same length. Figure 3.10 shows an example of DTW between two time series.

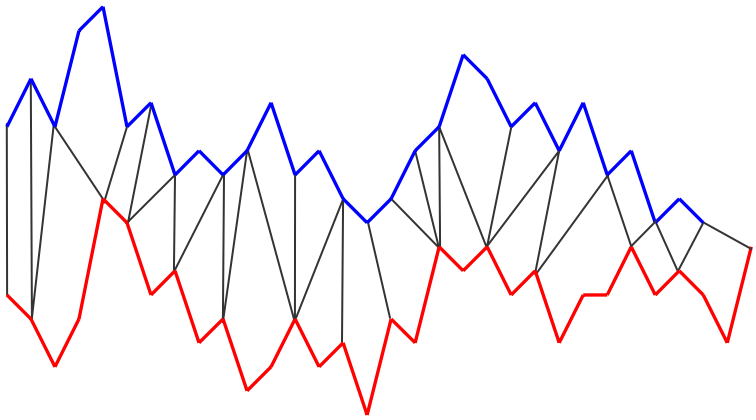


Fig. 3.10. Dynamic time warping

DTW is known to give better, or even much better, results than the Euclidean distance in several time series data mining tasks such as classification and clustering. Its applicability to time series of different lengths is another advantage that it has over the Euclidean distance. However, its main drawback is that its complexity $O(n^2)$ is very high compared with that of the Euclidean distance $O(n)$. Another disadvantage that DTW has compared with the Euclidean distance is that DTW violates the triangle inequality.

Several techniques have been proposed to index DTW some of which are approximate while others are exact. We present here some of the exact indexing methods, i.e. they guarantee no false dismissals.

In [213] a lower bounding measure was introduced. This lower bounding measure can be illustrated in Figure 3.11. We can see in that the sum of the squared lengths of the lines of shaded areas can be used as a lower bounding measure because this sum represents the minimum by which these points contribute to DTW.

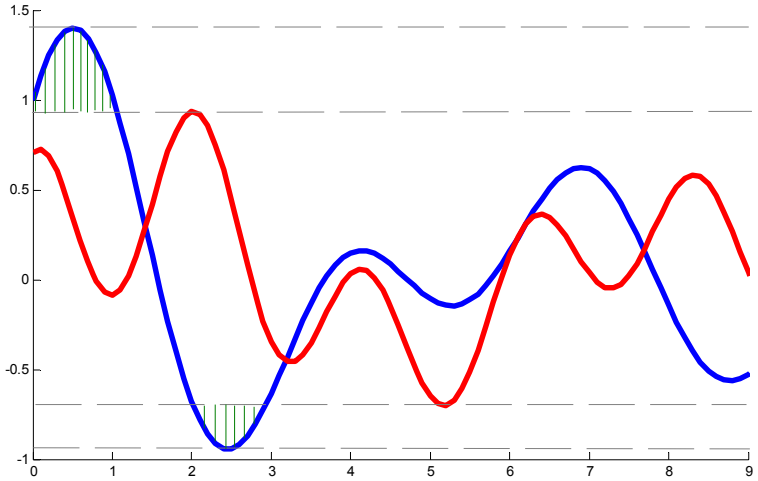


Fig. 3.11. The lower bounding measure presented in [213]

In [99] another lower bounding measure was introduced. This lower bounding measure is the maximum absolute difference of four values which are: the two first points of the two sequences , the two last points, the two minimum points, and the two maximum points (in the case of L_∞ , as in the original paper). These values are illustrated in Figure 3-12. In the case of L_2 the lower bounding measure is the sum of the squared differences of these values.

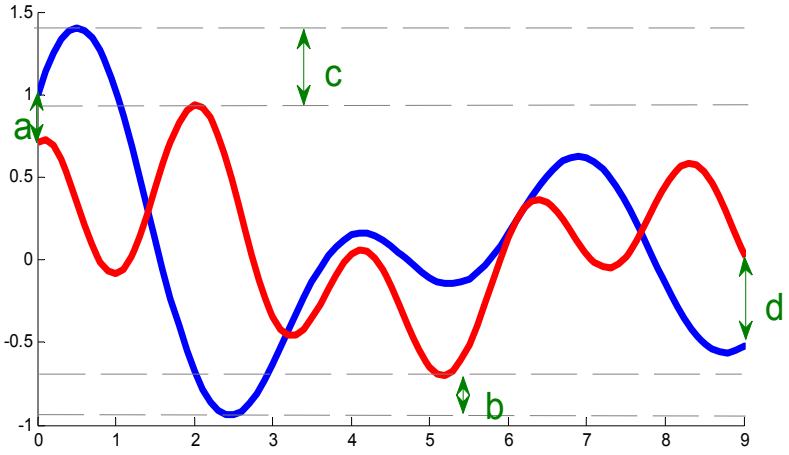


Fig. 3.12. The lower bounding measure presented in [99]

In [94] a new technique of DTW called LB_Keogh was proposed. This technique leans on two methods that constraint the indices of the warping path. These methods are Sakoe-Chiba Band and Itakura Parallelogram [158], [162]. These methods use a value r which defines the allowed warping path.

Given a sequence Q, by using r we can define two new sequences: U (upper), and L (lower) as follows:

$$U_i = max(q_{i-r} : q_{i+r}) \tag{3-25}$$

$$L_i = min(q_{i-r} : q_{i+r}) \tag{3-26}$$

From (2-25), (2-26) we can easily notice that:

$$U_i \geq q_i \geq L_i \qquad \forall i \tag{3-27}$$

Using the above concepts the LB_Keogh lower bounding between two sequences Q, C can be defined as:

$$LB\_Keogh(Q,C) = \sqrt{\sum_{i=1}^{n} \begin{cases} (c_i - U_i)^2 & if \quad c_i > U_i \\ (c_i - L_i)^2 & if \quad c_i < L_i \\ 0 & if \quad otherwise \end{cases}} \tag{3-28}$$

3.5.3 The Longest Common Subsequence (LCSS)

This distance was proposed mainly to handle noisy data [24], [196]. The LCSS is an elastic distance which allows the two sequences to stretch in order to match identical elements between them.

Given two sequences $S = \{s_1, s_2, ..., s_n\}$, $R = \{r_1, r_2, ..., r_m\}$, the LCSS between S and T can be defined as follows:

$$LCSS(S,R) = \begin{cases} 0 & if \quad S \quad or \quad R \quad is \quad empty \\ 1 + LCSS(Rest(S), Rest(R)) & if \quad s_1 = r_1 \\ max(LCSS(Rest(S), R), LCSS(R, Rest(S))) & otherwise \end{cases} \tag{3-29}$$

We can easily see from the above definition that LCSS does not follow the triangle inequality.

The LCSS can be solved using dynamic programming in $O(n^2)$ complexity [52]. Figure 3-13 shows the LCSS between two sequences.

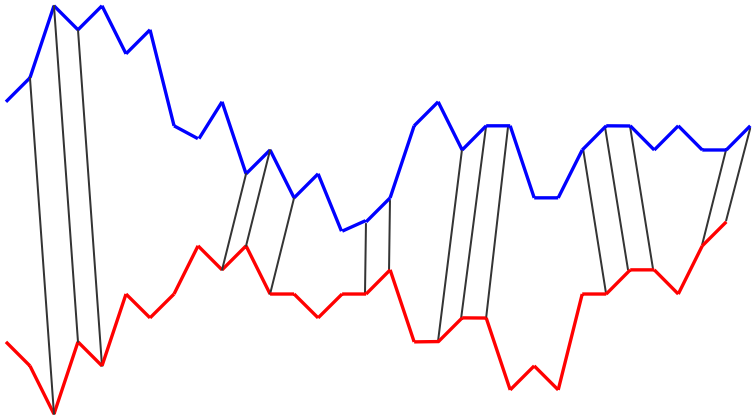


Fig. 3-13. The LCSS between two sequences.

LCSS as presented in (3-29) is mainly used to compare sequences of symbols which is not suitable in the case of time series with real values. In [195] the above model was extended to allow more flexible matching. This new measure is defined as follows:

$$LCSS_{\delta,\varepsilon}(S,R)=\begin{cases}0 & if\ \ S\ \ or\ \ R\ \ is\ \ empty\\ 1+LCSS_{\delta,\varepsilon}(Head(S),Head(R)) & if\ \ |s_n-r_n|<\varepsilon\ \ and\ \ |m-n|\le\delta\\ max(LCSS_{\delta,\varepsilon}(Head(S),R),LCSS_{\delta,\varepsilon}(R,Head(S))) & otherwise\end{cases} \tag{3-30}$$

Where $Head(S)=\{s_1,s_2,...,s_{n-1}\}$, $Head(R)=\{r_1,r_2,...,r_{m-1}\}$, δ is an integer and ε is a real positive number.

Based on (3-30) the authors of [195] propose two similarity function; the first is denoted by $d1$ and is defined as follows:

$$d1(\delta,\varepsilon,S,R)=\frac{LCSS_{\delta,\varepsilon}(S,R)}{min(m,n)} \tag{3-31}$$

The similarity measure defined in (3-31) is used to define another similarity measure $d2$;

Let F be a family of translations that cause vertical up or down shifts. A function f_C belongs to F if it has the form: $f_c(S)=(a_{x_1,1}+c,....,a_{x_1,n}+c)$. Based on this family of translations, the authors of [195] define another similarity measure:

$$d2(\delta,\varepsilon,S,R)=\max_{f_c\in F}\ d1(\delta,\varepsilon,S,f_c(R)) \tag{3-32}$$

This similarity measure is more flexible than $d1$ defined in (3-31).

3.5.4 The Edit Distance with Real Penalty (ERP)

This distance presented in [41] can be seen as a combination of the L_1 distance and the edit distance. It is defined as:

$$ERP(S,R)=\begin{cases}\sum_{1}^{M}|s_i-g| & if\quad N=0\\ \sum_{1}^{N}|r_i-g| & if\quad M=0\\ min\{ERP(Rest(S),Rest(R))+dist_{erp}(s_1,r_1),\\ \quad ERP(Rest(S),R)+dist_{erp}(s_1,g),\\ \quad ERP(S,Rest(R))+dist_{erp}(r_1,g)\} & otherwise\end{cases} \tag{3-33}$$

Where

$$dist_{erp}(s_i,t_I)=\begin{cases}|s_i-r_i| & if\quad s_i,r_i\quad are\quad not\quad gaps\\ |s_i-g| & if\quad t_i\quad is\quad a\quad gap\\ |r_i-g| & if\quad s_i\quad is\quad a\quad gap\end{cases}$$

and where M , N are the lengths of the time series S , R , respectively, and where g represents a gap (added symbol) which is assigned a constant value.

The authors of [42] prove that ERP follows the triangle inequality.

3.5.5 The Edit Distance on Real Sequences (EDR)

In order to handle data containing local time shifting and noise, the authors of ERP presented another distance in [43] which is called *Edit Distance on Real Sequences*. This distance can be defined as follows; given two sequences S , R of lengths M , N , respectively, EDR between S and T can be defined as:

$$EDR(S,R)=\begin{cases} M & if \quad N=0 \\ N & if \quad M=0 \\ \\ min\{EDR(Rest(S),Rest(R))+subcost, & \\ \quad EDR(Rest(S),R)+1, & \\ \quad EDR(S,Rest(R))+1\} & otherwise \end{cases} \tag{3-34}$$

where $subcost=0$ if $match(s_1,r_1)=true$ and $subcost=1, otherwise$, and where two elements s_i, r_j from S,T , respectively, are said to *match* if and only if $|s_{i,x}-r_{j,x}|\le\varepsilon$ and $|s_{i,y}-r_{j,y}|\le\varepsilon$, where ε is the *matching threshold*.

3.5.6 Dissimilarity Distance (DISSIM)

This distance was presented in [69] and it calculates the similarity between two sequences with different sampling rates. DISSIM is defined between two sequences S,R as the definite integral of the function of time of the Euclidean distance between the two sequences, i.e.:

$$DISSIM(S,R)=\int_{t_1}^{t_n} D_{S,R}(t).dt \tag{3-35}$$

where $D_{S,R}(t)$ is the function of the Euclidean distance. Because of the discrete nature of the sequences, the above definition can be written as:

$$DISSIM(S,R) = \sum_{k=1}^{n-1} \int_{t_k}^{t_{k+1}} D_{S,R}(t).dt \tag{3-36}$$

It is easy to notice that DISSIM as presented in the above relation is too difficult to calculate. The authors, however, present an approximation to this distance using the following expression:

$$DISSIM(S,R) \approx \frac{1}{2} \sum_{k=}^{n-1} \left(\left(D_{S,R}(t_k) + D_{S,R}(t_{k+1}) \right) . (t_{k+1} - t_k) \right) \tag{3-37}$$

3.5.7 Similarity Search based on Threshold Queries (TQ)

This method proposed in [12] uses a novel concept called *Threshold Queries*. Given a time series $S = \{(s_i, t_i) \in \mathsf{R} \times \mathsf{T}, i = 1,2,...,n\}$, and a parameter $\tau \in \mathsf{R}$ that represents a *threshold* then the *threshold-crossing time interval* of *S* with respect to τ is a sequence:

$$TCT_\tau(S) = \{(l_j, u_j) \in \mathsf{T} \times \mathsf{T};\ j = \{1,2,...,m\}, m \le n\} \tag{3-38}$$

(*l* and *u* stand for *lower* and *upper*, respectively) of time intervals, such that :

$$\forall t \in \mathsf{T} : (\exists \quad j \in \{1,2,...,m\}; l_j < t < u_j) \Leftrightarrow s(t) > \tau \tag{3-39}$$

An interval $tct_{\tau,j} = (l_j, u_j)$ of $TCT_\tau(S)$ is called *threshold-crossing time interval.*

In simple words, TQ transforms the time series into intervals where all the points within each interval are greater than the given parameter τ, and each interval is transformed into a 2-dimensional space, where the start point and the end point of each interval represent the two dimensions.

The distance between two time intervals is defined as follows: given two time intervals $t1 = (t1_l, t1_u)$, $t2 = (t2_l, t2_u)$, then the distance between these two intervals is defined as :

$$d_{int}(t1,t2) = \sqrt{(t1_l - t2_l)^2 + (t1_u - t2_u)^2} \tag{3-40}$$

That is to say two intervals are considered *similar* if they are starting at *similar* starting points and ending at *similar* ending points.

3.5.8 Spatial Assembling Distance (SpADe)

[46] proposed a new warping distance which is able to handle shifting and scaling in both temporal and amplitude dimensions. The main idea of their work is to use a sliding window of a fixed size to extract a set of small patterns from time series. These patterns are called *local patterns*, so a local pattern of a time series S can be defined as $l_p = (\theta_{pos}, \theta_{amp}, \theta_{shp}, \theta_{tscl}, \theta_{ascl})$, which represent the position of l_p in S, the mean amplitude of the data in l_p, the shape signature, the temporal scale, and the amplitude scale of l_p, respectively, and where $\theta_{tscl}, \theta_{ascl}$ are given a value equal to 1 if S is not scaled. The distance between two local patterns l_p in S and $l_p^{'}$ in R is defined as follows:

$$D_1(l_p, l_p^{'}) = f\left(\left|\theta_{amp} - \theta_{amp}^{'}\right|, \left|\theta_{shp} - \theta_{shp}^{'}\right|\right) \tag{3-41}$$

where f is a weight vector and it is application-specific.

If $D_1(l_p, l_p^{'}) \leq \varepsilon$ then we have a *local pattern match* LPM. The problem of similarity is thus transformed to finding the most similar set of matching patterns.

3.5.9 Sequence Weighted Alignment (Swale)

The Swale scoring model presented in [142] uses a similarity score that rewards matches and penalizes mismatches. It also allows the match reward and the gap penalties to be weighted relatively to one another.

Given two time series S whose length is n and R whose length is m. Let the gap cost be gap_c and the match reward be $reward_m$, then:

$$Swale(S,R) = \begin{cases} m.gap_c & if \quad n = 0 \\ n.gap_c & if \quad m = 0 \\ reward_m + Swale(Rest(S), Rest(R)) & if \quad \forall d, \left|s_{d,1} - r_{d,1}\right| \leq \varepsilon \\ max\{gap_c + Swale(Rest(S), R), \\ \quad gap_c + Swale(S, Rest(R))\} & otherwise \end{cases} \tag{3-42}$$

where d is the dimension.

3.6 Summary

In this chapter we introduced fundamental concepts in data mining in addition to the principal problems this branch of computer science handles. We mainly focused on time series, a ubiquitous data type that occurs in many fields of science, medicine as well as economics. We presented the different versions of the similarity search problem in time series databases and showed why this problem is difficult to manage.

In the second part we presented the GIMINI algorithm which offers a frame to process the similarity search in time series information retrieval. This algorithm is based on lowering the high dimensionality of the time series data and processing the query in the reduced spaces under certain conditions, to guarantee no false dismissals, to obtain a candidate response set that is later post-processed to return the final response set.

Next, we did a survey of the most known dimensionality reduction techniques in time series information retrieval. We presented a description of the basis of each one, in addition to its main characteristics and how it approaches the questions of dimensionality reduction and the no false dismissal property. A comparison between the different reduction techniques based on the theoretical characteristics of each representation method was made. We showed how an experimental comparison may give different results.

In the following part of the chapter we presented different distance measures. We presented the measures which are widely used in time series information retrieval. We also presented other measures, like LCSS, which is applied to other data types too. We described each measure and showed its advantages and disadvantages in addition to its different variations and the modification that were made to improve it.

In the last part of this chapter we presented the state-of-the-arts distance measures, described these new distance measures, and showed their advantages.

Chapter 4

Adaptive Sampling and Time Series Classification

In this short chapter we present our first contribution of this dissertation [122], which is an experimental study to evaluate the impact of dimensionality reduction on a time series classification task. This dimensionality reduction is achieved using an adaptive sampling technique of the time series. The experiments utilize different compression ratios and different similarity measures and metric distances.

We like to state that the work presented in this chapter is preliminary. We present it mainly because it is a part of the work we did in this dissertation. But we realize that it requires further development. Besides, the experiments we conducted concern only a synthetic dataset, so we do not claim that the results can be extended to other datasets.

We start by presenting the necessary background in section 4.1. The proposed method is introduced in section 4.2. Section 4.3 is the experimental section. In section 4.4 we terminate this chapter with concluding remarks and directions of future work.

4.1 Introduction and Related Work

In the previous chapter we covered some dimensionality reduction techniques in time series information retrieval. Dimensionality reduction can also be viewed as a *data compression* technique. Data compression refers to the process of reducing the size of a data file. Original representation of data has redundancies and compressing the data reduces these redundancies [163].

Compression techniques belong to one of two classes [33]: *lossless* compression, in which the original data can exactly be reconstructed from the compressed data, and *lossy* compression, which allows an approximate reconstruction of the original data.

The performance of compression methods is evaluated using different measures. One of them is the *compression ratio* [164] which is defined as:

$$Compression\ \ ratio = \frac{size\ \ of\ \ the\ \ output\ \ stream}{size\ \ of\ \ the\ \ input\ \ stream} \tag{4-1}$$

In the language of time series information retrieval and data mining the above formula can be expressed as:

$$Compression\ \ ratio = \frac{dimension\ \ of\ \ the\ \ reduced\ \ space}{dimension\ \ of\ \ the\ \ original\ \ space} \tag{4-2}$$

In many applications the amount of available data is enormous that storage, transmission, and computation display serious challenges that the need for data compression techniques becomes vital [127]. In sensor networks, for instance, there are energy and bandwidth limitations, so it is desirable to compress time series to meet these limitations [41].

In some applications in time series data mining low variability of time series-signal profiles is a very important property. In these cases it is desirable to find a compression algorithm that could benefit from the low variability among signal profiles [62].

Several papers have addressed the problem of time series classification as one of the main tasks of time series data mining. In [97] the authors presented the *Cluster, Then Classify* method. In this method every class is represented and a piecewise linear representation that includes weights per segment is proposed [140]. The same representation was used in [72] but the classification rules were induces with decision trees. In [216] the authors use *Hidden Markov Model* (HMM) representations to classify time series. In [144] the authors use neural networks that are trained on statistical features. The authors of [7] propose a method capable of providing a classification even when only a part of the time series is available. In this method the induced classifiers are a linear combination of literals. This combination is obtained by boosting base classifiers that contain one literal only. In [85] the authors tackle this problem by assigning a weight to each training instance. This weight is used in the generalization phase to calculate the distance between a query and that instance. The approach presented in [204] is based on training the distance metric so that the *k*-nearest neighbors always belong to the same class while examples from different classes are separated by a large margin.

4.2 Adaptive Sampling of Multidimensional Curves

The proposed method is inspired by the *Dynamic Programming Piecewise Linear Approximation* (DPPLA) model presented in [123], [121], and derived from [152] and [102]. The authors propose a data modeling approach to handle adaptive sampling based on a suboptimal solution that limits the search space of a dynamic programming solution to the problem of polygonal curve approximation.

Given an *n*-dimensional time series S, we are looking for an approximation $S_{\hat{\theta}}$ to S so that :

$$\theta = \underset{\theta}{ArgMin}\left(E\left(S, S_\theta\right)\right) \tag{4-3}$$

where E is the root mean square error between S and the model S_θ. The search of the family $\{S_\theta(n)\}$ is limited to the set of linear piecewise functions, and is also restricted to the case where the two end points of the segments lie on S. In this case θ is the set of discrete time locations $\{n_i\}$ and the end point of a segment is the start point of another. The selection of the optimal set of parameters $\hat{\theta} = \{\hat{n}_i\}$ is performed using dynamic programming as follows:

First, we define the compression ratio of the piecewise approximation as:

$$\rho = 1 - \frac{\left|\{n_i\}\right|}{\left|\{S(n)\}\right|} \times \frac{\rho + 1}{\rho} \tag{4-4}$$

where $S(n) \in \mathsf{R}^n, \forall n$.

Given a value of ρ and the length of the time window $w = \left|\{S(n)\}_{n \in \{1,\ldots,w\}}\right|$, the number $N = \left|\{n_i\}\right| - 1$ of piecewise linear segments is known.

Let $\theta(k)$ be, by definition, the parameters of a piecewise approximation containing k segments, let $\delta(k,i)$ be the minimal error of the best piecewise linear approximation containing k segments and covering the time window $\{1,\ldots,,w\}$, then $\delta(k,i)$ can be expressed as:

$$\delta(k,i) = \underset{\theta(k)}{Min}\left\{\sum_{n=1}^{i}\left\|S_{\theta(k)}(n) - S(n)\right\|^2\right\} \tag{4-5}$$

The above term, according to Bellman optimality principle [16], can be decomposed as:

$$\delta(k,i) = \underset{k-1 \le n_k \le i}{Min}\left\{d\left(n_k, i\right) + \delta\left(k-1, n_k\right)\right\} \tag{4-6}$$

where $d(n_k, i) = \sum_{n=n_k}^{i}\left\|R_{k,i}(n) - S(n)\right\|^2$ and $R_{k,i} = \left(S(i) - S(n_k)\right).\frac{n - n_k}{i - n_k} + S(n_k)$ is the linear segment between $S(i)$ and $S(n_k)$.

The recursion is initialized by observing that:

$$\delta(k,i) = 0 \qquad ,\forall k, \forall i < k \tag{4-7}$$

At the end of the recursion we get the optimal piecewise linear approximation, i.e. the set of time locations of the end points of the linear segments:

$$\hat{\theta}(k) = \underset{\theta(k)}{ArgMin}\left\{ \sum_{n=1}^{w} \left\| S_{\theta(k)}(n) - S(n) \right\|^2 \right\} \tag{4-8}$$

with the minimal error:

$$\delta(k,w) = \sum_{n=1}^{w} \left\| S_{\hat{\theta}(k)}(n) - S(n) \right\|^2 \tag{4-9}$$

The complexity of this algorithm is $O(k.w)$. In order to reduce this complexity the search window can be limited by using a lower bound $lb = max\{i - band, 0\}$ for each step i, and where $band$ is a user-defined parameter:

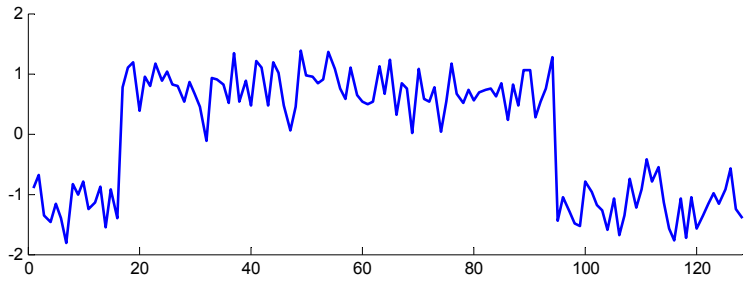
$$\delta(k,i) = \underset{lb \le n_k \le i}{Min} \{ d(n_k, i) + \delta(k-1, n_k) \} \tag{4-10}$$

In practice we choose $band = 2.w/k$.

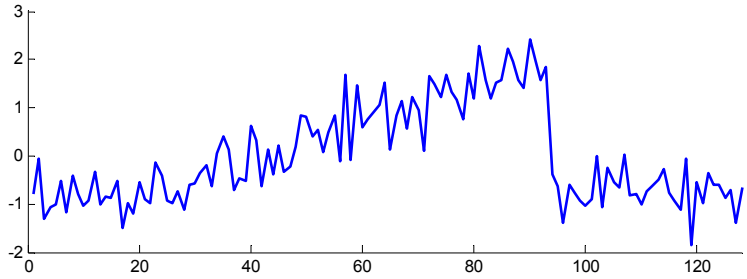
4.3 The Experiments

4.3.1 The Dataset

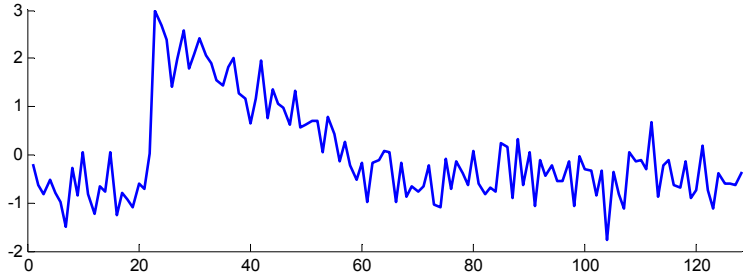
The dataset on which we conducted the experiments is called “cylinder-bell-funnel”. This dataset widely-used in the time series data mining community was originally introduced by [165] and further worked on by [120] and later by [94]. The learning task is to distinguish between three classes: *cylinder* (*c*), *bell* (*b*), and *funnel* (*f*). Figure 4.1 shows examples of the three classes.



cylinder



bell



funnel

Fig. 4.1. Examples of the three classes of dataset "cylinder-bell-funnel"

The three classes are generated using the following functions:

$$
\begin{aligned}
c(t) &= (6+\eta).\chi_{[a,b]}(t) + \varepsilon(t) \\
b(t) &= (6+\eta).\chi_{[a,b]}(t).\frac{t-a}{b-a} + \varepsilon(t) \\
f(t) &= (6+\eta).\chi_{[a,b]}(t).\frac{b-t}{b-a} + \varepsilon(t)
\end{aligned}
\qquad (4\text{-}11)
$$

where :

$$
\chi_{[a,b]} = \begin{cases} 0 & if \quad t < a \vee t > b \\ 1 & if \quad a \le t \le b \end{cases}
$$

and where η and $\varepsilon(t)$ are drawn from a standard normal distribution $N(0,1)$, a is an integer drawn from a uniform distribution in the interval $[16,32]$ and $b-a$ is another integer drawn from another uniform distribution in the interval $[32,96]$. In Figure 4.2 we see another example of the three classes of this dataset with their corresponding approximations obtained by adaptive sampling.

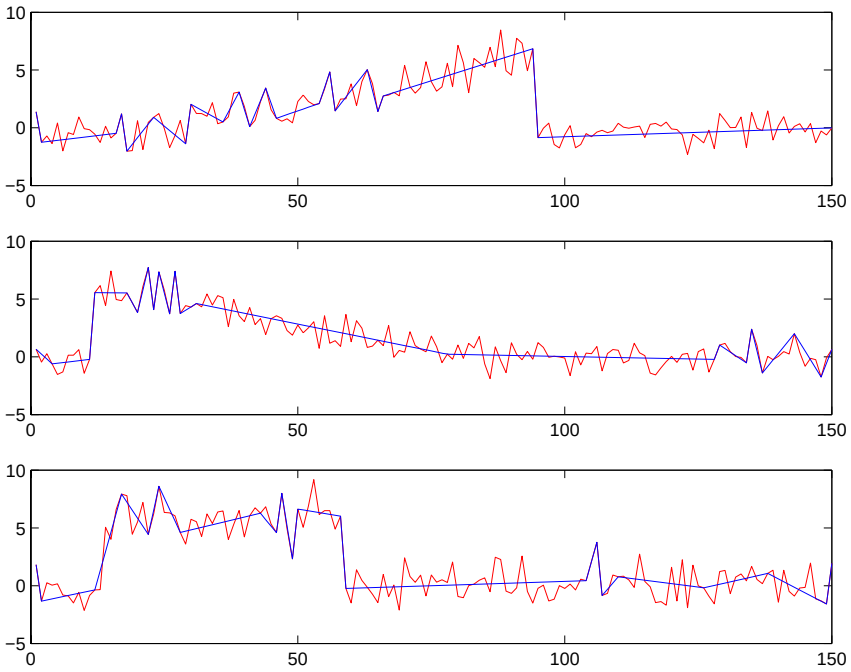


Fig. 4.2. Examples of the three classes of dataset "cylinder-bell-funnel" together with their corresponding approximation with an 87% compression ratio

4.3.2 The Experimental Protocol

20 time series of 9 shapes are generated. For each series 3 shapes belong to class (*c*) , 3 belong to class (*b*) , and 3 belong to class (*f*). We tested our method in a classification task based on the first near-neighbor rule using leaving-one-out cross validation. This means that every curve is compared to the other 8. If the 1-NN does not belong to the same class, the error counter is incremented by 1.

For each of the tested methods, the mean and the standard deviation on all the 20 time series are computed.

4.3.3 The Results

Table 4.1 shows a comparison of the three methods: adaptive sampling, piecewise linear approximation (PLA) and discrete Fourier transform (DFT). In the case of PLA the Euclidean distance (ED-PLA), in addition to dynamic time warping (DTW-PLA), are used.

The results show that DTW is particularly adapted to the classification task in question. A gain of 57% compared with ED is obtained , and 35% compared with DFT .We see from the table that using adaptive sampling associated with DTW (DTW-PLA) only slightly degrades the performance of the original data with DTW .

We also notice that for a compression ratio of 25% or less, the results are comparable. For compression ratios of the order of 75%, the loss is only of the order of 5% on average. We also see from the table that the performance does not degrade substantially except for compression ratios of the order of 90%. The compression method using DFT seems to be less sensitive to the compression ratio and yields error rates that vary between 30% and 45%. When the latter is associated with adaptive sampling, the performance degrades.

Table 4.1. Comparison of the mean error rate and the standard deviation of PLA-DTW, PLA-ED, and DFT

The Method	The Mean Error Rate	Standard Deviation
DTW $\rho = 0\%$	0.0056	0.0248
DTW-PLA $\rho = 5\%$	0.0056	0.0248
DTW-PLA $\rho = 10\%$	0.017	0.041
DTW-PLA $\rho = 25\%$	**0.00**	**0.00**
DTW-PLA $\rho = 50\%$	0.028	0.049
DTW-PLA $\rho = 75\%$	0.05	0.092
DTW-PLA $\rho = 90\%$	0.328	0.175
ED $\rho = 0\%$	0.578	0.133
ED-PLA $\rho = 5\%$	0.161	0.132
ED-PLA $\rho = 10\%$	0.228	0.117
ED-PLA $\rho = 25\%$	0.156	0.115
ED-PLA $\rho = 50\%$	0.205	0.166
ED-PLA $\rho = 75\%$	0.333	0.184
ED-PLA $\rho = 90\%$	0.583	0.213
DFT $\rho = 0\%$	0.311	0.152
DFT $\rho = 5\%$	0.372	0.158
DFT $\rho = 10\%$	0.433	0.129
DFT $\rho = 25\%$	0.406	0.181
DFT $\rho = 50\%$	0.317	0.174
DFT $\rho = 75\%$	0.322	0.200
DFT $\rho = 90\%$	0.300	0.191
DFT-PLA $\rho = 5\%$	0.478	0.188
DFT-PLA $\rho = 10\%$	0.422	0.164
DFT-PLA $\rho = 25\%$	0.450	0.175
DFT-PLA $\rho = 50\%$	0.511	0.174
DFT-PLA $\rho = 75\%$	0.533	0.203
DFT-PLA $\rho = 90\%$	0.638	0.210

4.4 Conclusion

In this chapter we presented the first contribution of this dissertation, which is an experimental study of using adaptive sampling as a method of dimensionality reduction in a time series classification task. The results of the experiments we presented in this chapter on this synthetic dataset show that even when using high compression ratios, the performance of the adaptive sampling method is still acceptable in a classification task. These results are obtained when DTW is used as a similarity measure. The results also show that for a low compression ratio, the

performance seems to be better than classifying the raw data with DTW. We think the reason for this is that adaptive sampling has a smoothing effect.

When comparing this method with DFT the proposed method shows an improvement in performance of 15%-20%.

However, as indicated in the introduction, this study is preliminary and it requires further investigation. Besides, the results were obtained using one dataset so they can not be generalized to other datasets.

4.5 What Next?

We think that the most important improvement that we can investigate is to restrict the sampling so that some points of the sample will be in the neighbored of some important points in the original time series. This approach can be viewed as combining our method with landmark methods.

The proposed method can also be combined with the concept of multi-resolution we are going to present in Chapter 6. The idea is to have several classifications that are related to different compression ratios of adaptive sampling in order to avoid the problem of over-fitting on the training set. In other words, we can use different compression ratios that correspond to “coarser” through “finer” classifications, and later a validation set chooses the best compression ratio for the classification task.

Chapter 5

Symbolic Methods in Time Series Information Retrieval

In this chapter we introduce our contributions in two principal directions: the first is our work on improving the edit distance which we presented in [129], [133], [134], [135], and [137]. We also present the results of another paper that was accepted, but we decided not to publish it (see the introduction of this dissertation) about the PFEED distance presented in Section 5.4.

The basis of these improvements is to give the edit distance the ability to explore global similarity in addition to local one. Extensive experiments are also presented to show the advantages of our improved versions of the edit distance, and to investigate how these improvements can be developed to handle other problems in time series data mining.

The second principal direction of this chapter is a new minimum distance on which SAX, one of the most effective symbolic methods, is based, which is the work we presented in [128]. We show the results of our experiments that illustrate how this new minimum distance enhances the performance of SAX. We start this chapter by presenting in Section 5.1 some time series symbolic representation methods focusing on SAX. In Section 5.2 we present EED our first extension of the edit distance. The second extension MREED is presented in Section 5.3. The general model of extension Σ_{GRAM} is presented in Section 5.4. PFEED, a non-parametric extension of the edit distance is presented in Section 5.5. Section 5.6 is the second part of this chapter where we introduce UMD, a new minimum distance for SAX. Section 5.7 is the concluding section and in Sections 5.8 we present some directions of future work.

We have to mention that presenting the results of our experiments in this chapter was a bit problematic. In the papers we published we compared the results of different methods by comparing the mean and the standard deviation of the compared methods over all the data sets. We also added another comparison criterion which is the number of times a particular method outperformed another one. However, later we were wondering if this is the correct way to present the results. On the one hand, taking the mean and the standard deviation on all the datasets, whose nature is different, did not seem statistically correct. On the other hand, counting the number of times one method outperformed another oversimplifies the comparison. Because if a method A outperforms another method B by hardly 0.1 % on one dataset, while method B outperforms B by almost 20 % on another dataset, we can not say that the two methods have the same performance.

The way this comparison is made in most papers is by showing, mainly graphically, the comparison results of few datasets, but this was not practical in our case because we conducted extensive experiments that tables were the only proper way to show the results.

So we finally decided to just highlight the best result for each dataset and let the reader conclude which method is best according to how many datasets a certain method outperformed another but also by how much it outperformed it.

5.1 Introduction to Symbolic Representation

Among dimensionality reduction techniques that we presented in Chapter 3, symbolic representation of time series has several advantages which interested researchers in this field of computer science for long, because symbolic representation permits benefiting from the ample symbolic algorithms known in the text-retrieval and bioinformatics communities.

5.1.1 Symbolic Representation Scheme

Symbolic representation of time series uses an alphabet $\sum$ (usually finite) to reduce the dimensionality of the time series. This can be defined formally as follows: given a time series $S = \{(t_1, s_1), (t_2, s_2), ..., (t_n, s_n)\}$, the symbolic representation scheme is a map:

$$[t_i, t_j] \xrightarrow{f} \alpha_k \quad ; \alpha_k \in \sum \quad i < j \tag{5-1}$$

Numeric time series can be converted into symbols through a process called *discretization* or *symbolization*. This discretization greatly increases the efficiency of numerical computations compared with what it would be with the original raw data [54].

We personally think that discretization is an intrinsic part of time series data because the phenomenon in question is usually continuous, but we measure it at discrete time stamps (regular or irregular), which is in fact a form of quantization or discretization.

Transforming raw data into symbols can be achieved using different techniques. The TREND method [25], for instance, integrates signal processing and natural language description to automatically generate textual description of time series data. The basis of TREND is to use wavelets and space scale theory to detect trends in the time series. These trends can be short-term or long-term trends.

The idea of using language based description of the time series was also exploited in [178]. The method presented in that paper consists of three steps:

1- Selecting the important trends and patterns that need to be communicated.
2- Mapping these patterns and trends onto words and phrases.
3- Generating actual texts which are based on these words and phrases.

In [9] the authors use an alphabet which they call *The Shape Description Alphabet* (SDA) to encode the shape of the time series. However, this method is similar to the language based methods in that the characters actually describe trends in the time series.

In [82] the authors present a method called *Interactive Matching of Patterns with Advanced Constraints in Time-Series databases* (IMPACTS). This method uses a string based approach to discretize the time series. The string is built by using change ratios between two consecutive time points. Next, the resulting string can be indexed using a suffix tree which indexes all the strings in the database.

The authors of [53] propose a rule-discovery based technique to symbolize the time series. The basis of their technique is to cluster related patterns in the sliding window and to assign a symbol to that cluster. Their focus is on local patterns rather than global ones.

Clustering was also exploited by [77], [78] to find the shapes that occur frequently. Their work focuses on compressing long time series into symbols. The resulting symbolic representation is then used for visualization, manual edition, and mining for sequential patterns.

In general, most of these symbolic representations methods suffered from two main inconveniences [112]: the first is that the dimensionality of the symbolic representation method is the same as that of the original space, so there is no virtual dimensionality reduction. The second drawback is that although distance measures have been defined on the reduced symbolic spaces, these distance measures are poorly correlated with the original distance measures defined on the original spaces.

There are several distance measures that apply to symbolic data. In the beginning these measures were restricted to data types whose representation is naturally symbolic (DNA and proteins sequences, textual data, etc). But later these symbolic measures were also applied to other data types that can be transformed into strings by using symbolic representation techniques.

Of all the symbolic representation methods in the times series data mining literature, the *Symbolic Aggregate approXimation* method (SAX) [111] stands out as probably the most powerful symbolic representation method. The main advantage of SAX is that the similarity measure it uses, called MINDIST, uses statistical lookup tables, which makes it is easy to compute with an overall complexity of $O(N)$.

Due to its importance, SAX will be described in detail in the following section.

5.1.2 Symbolic Aggregate Approximation (SAX)

The main characteristic of SAX is that it is similar in structure to the other representation methods we presented in Chapter 3. This means it permits dimensionality reduction and it also presents a similarity measure which lower bounds the original distance in the original space.

SAX is based on the fact that normalized time series have highly Gaussian distribution [108], so by determining the breakpoints that correspond to the alphabet size, one can obtain equal sized areas under the Gaussian curve.

SAX is applied as follows:

1- The time series are normalized.

2- The dimensionality of the time series is reduced by using PAA.

3- The PAA representation of the time series is discretized. This is achieved by determining the number and location of the breakpoints. The number of breakpoints is related to the desired alphabet size (which is chosen by the user), i.e. *alphabet_size=number_of_breakpoints+1*. Their locations are determined by statistical lookup tables so that these breakpoints produce equal-sized areas under the Gaussian curve.

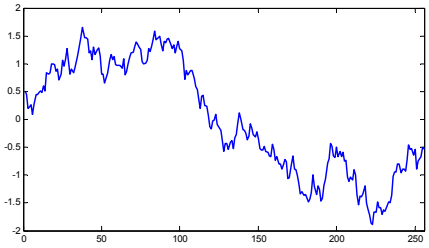
The interval between two successive breakpoints is assigned to a symbol of the alphabet, and each segment of the PAA that lies within that interval is discretized by that symbol. The last step of SAX is using the following similarity measure:

$$MINDIST\left(\hat{S},\hat{R}\right)\equiv\sqrt{\frac{n}{N}}\sqrt{\sum_{i=1}^{N}\left(dist\left(\hat{s}_i,\ \hat{r}_i\right)\right)^2} \qquad (5\text{-}2)$$

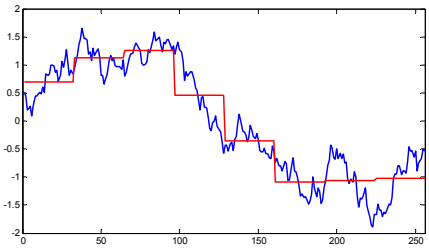
Where n is the length of the original time series, N is the length of the strings (the number of the frames), $\hat{S}$ and $\hat{R}$ are the symbolic representations of the two time series S and R, respectively, and where the function $dist(\ )$ is implemented by using the appropriate lookup table.

Notice that MINDIST is a similarity measure and not a distance metric since it violates two conditions of distance metrics which are the identity condition, and the triangle inequality condition, and it respects only one condition which is the symmetry condition.

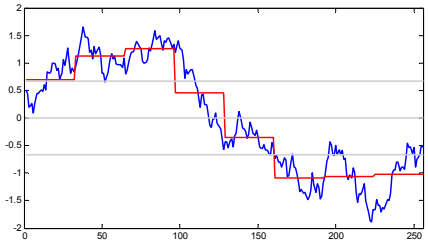
(**a**) The original time series



(**b**) Converting the time series to PAA



(**c**) Choosing the breakpoints



(**d**) Discretizing the PAA

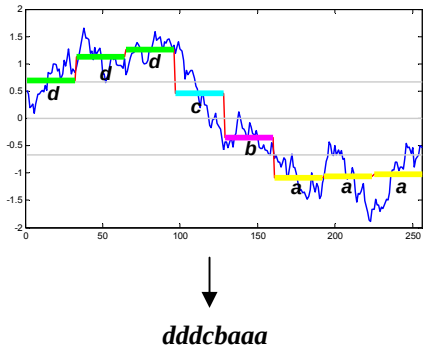


Fig. 5.1. The different steps of SAX

It can easily be proven that MINDIST is a lower bounding of the Euclidean distance, because it is a lower bounding of the similarity measure used in PAA. This guarantees no false dismissals. Figure 5.1 illustrates the different steps of SAX. In this example the time series is transformed from a *256-dimension* space, into an *8-dimension* space. The alphabet size is 4.

SAX paved the way for other papers in time series data mining, time series information retrieval, and other related fields in computer science. In [96] the authors presented HOT SAX, which is an algorithm of finding time series *discords*; subsequences of a longer time series that are maximally different to all the rest of the time series subsequences. These discords have many applications in time series data mining.

In [203] the problem of discords was further extended to handle images. The algorithm presented in that paper finds *shape discords*, which are the most unusual shapes in a collection of data. The algorithm uses locality-sensitive hashing to estimate similarity between pairs of shapes and then generates heuristics to reorder the search more efficiently

Another extension of SAX called the *indexable Symbolic Aggregate approXimation* (*i*SAX) was presented in [173] and [174]. The main objective of this method is to index massive datasets. The basis of *i*SAX is to modify SAX to allow extensible hashing. This modification permits indexing time series with zero overlap at leaf nodes which makes *i*SAX very fast.

The most recent extension that we know of is *i*SAX 2.0 [32] which is a data structure designed to index even larger collections of time series. The authors conduct experiments using one billion time series, which, according to the authors, is the largest dataset ever experimented on multimedia objects.

However, the version of SAX that we presented earlier, which is called *classic* SAX by the authors of all these extensions, is the most widely known one.

5.2 The Extended Edit Distance (EED)

5.2.1 Introduction

Strings, also called *sequences* or *words*, are a way of representing data. This data type exists in many fields of computer science such as molecular biology, where DNA (deoxyribonucleic acid) sequences are represented using four nucleotides which correspond to the four bases: adenine (A), cytosine (C), guanine (G) and thymine (T). This can be expressed as a 4-symbol alphabet. The RNA (ribonucleic acid) can also be expressed using a 4-symbol alphabet which corresponds to adenine (A), cytosine

(C), guanine (G) or uracil (U). Protein sequences, on the other hand, can be represented using a 20-symbol alphabet which corresponds to the 20 amino acids.

Written languages are also expressed in terms of alphabets with letters (26 in English). Spoken languages are represented using phonemes (40 in English). Texts use alphabets with a very large size (the vocabulary items of a language). Images can be expressed using an alphabet of three basic colors $\{r,g,b\}$. So we see from these examples that strings are ubiquitous.

As defined in Chapter 2, the edit distance (ED) is the minimum number of delete, insert, and substitute operations needed to transform string S into string T. This distance is the main distance measure used to compare two strings. The edit distance uses three atomic operations: insert, delete, and change. Figure 5.2 shows the edit distance between the two strings $S_1 = \{G,D,H,E,Y\}$ and $S_2 = \{S,H,Q,Y,G\}$

		G	D	H	E	Y
	0	1	2	3	4	5
S	1	1	2	3	4	5
H	2	2	2	2	3	4
Q	3	3	3	3	3	4
Y	4	4	4	4	4	3
G	5	4	5	5	5	4

Fig. 5.2. The edit distance between two strings.

The edit distance has a main drawback: it penalizes all change operations in the same way without taking into account the character that is used in the change operation. This drawback is due to the fact that the edit distance is a measure of local similarities in which matches between substrings are highly dependent on their positions in the strings [106]. In fact, the edit distance is based on local procedures both in the way it is defined and also in the algorithms used to compute it. This raises questions on the accuracy of the similarities obtained by applying this distance. We will give the following example to show this idea.

Example 5.1

The edit distance was presented mainly to apply to spelling errors. But because of the conventional keyboard arrangement, the probability that an "A" be mistyped as "Z" is not the same as mistyping "A" as "P", for instance (on an English keyboard), but yet, the edit distance does not take these different possibilities into consideration.

One of the ways that can be considered to deal with this problem is to use a predefined table that shows the cost of change between any two characters of that alphabet. This method, although worth considering, has a few cons; first, it is specific to each alphabet. Second, the number of change costs to be defined beforehand (this number is C_2^n, where n is the size of the alphabet) can be somehow large. Third, if we try to use multi-resolution techniques on the symbolic representation, we will have to define a table for each resolution. Another serious problem arises in this latter case which is the fact that merging two characters in text processing is not intuitional. So there is no clear way on how the "new" characters (those of a different resolution) can be related to the old ones. □

In this section, we present a new distance metric for symbolically represented data. The proposed metric, which we call the *Extended Edit Distance* (EED), adds a global feature of similarity to the ordinary edit distance. This feature enables our proposed distance metric to deal with the atomic edit operations more naturally because there is no need to predefine a cost function for the different operations and the algorithm assigns costs online based on the characters of the two compared strings. This proposed distance can by itself detect if the atomic edit operations are applied to characters that are "familiar" or "unfamiliar" to the two strings concerned.

Before we introduce our distance we start by giving this definition which is necessary to understand the new distance.

Definition-The Number of Distinct Characters (NDC): Given two strings S, T. The number of distinct characters NDC is defined as:

$$NDC(S,T) = \left|\{ch(S)\} \cup \{ch(T)\}\right| \tag{5-3}$$

where ch ( ) is the set of characters that a string contains.

5.2.2 Motivation

The following example shows the limitation of the edit distance.

Example 5.2

Given the string:

$$S_1 = marwan$$

By performing two change operations on S_1 in the first and fifth positions we obtain the string:

$$S_2 = aarwin$$

By calculating their edit distance we get:

$$ED(S_1, S_2) = 2$$

If we calculate their *NDC* we get:

$$NC(S_1, S_2) = 6$$

Now if we change the same positions in S_1 with different characters we obtain, for instance, the string:

$$S_3 = barwen$$

By calculating the edit distance between S_1 and S_3 we obtain:

$$ED(S_1, S_3) = 2 \text{ (which is the same as } ED(S_1, S_2)\text{)}$$

Now, if we calculate their *NDC* we get:

$$NC(S_1, S_3) = 7$$

This means that one change operation used a character that is more "familiar" to the two strings in the first case than in the second case, in other words, S_2 is closer to S_1 than S_3. However, the edit distance was not able to recognize this, since the edit distance was the same in both cases.

We will see later that this concept of "familiarity" can be extended to consider not only *NDC* but the frequency of subsequences too.

5.2.3 Definition-The Extended Edit Distance (EED)

Let Σ be a finite alphabet, and let Σ^* be the set of strings on Σ. Let $f_a^{(S)}$ be the frequency of the character a in S, and $f_a^{(T)}$ be the frequency of the character a in T,

where S, T are two strings in Σ^*. *The Extended Edit Distance* (EED) is defined as:

$$EED(S,T) = ED(S,T) + \lambda\left[|S| + |T| - 2\sum_{a\in\Sigma} min(f_a^{(S)}, f_a^{(T)})\right] \quad (5\text{-}4)$$

Where $|S|, |T|$ are the lengths of the two strings S, T respectively, and where $\lambda \geq 0$ ($\lambda \in \mathsf{R}$). We call λ the frequency factor.

Notice that when $\lambda = 0 \Rightarrow EED(S,T) = ED(S,T)$, and this is the minimum value for EED, so ED is actually a lower bound of EED.

5.2.4 Theorem 1

EED is a metric distance □

Before we prove the theorem we present the following lemma.

Lemma 1

$\forall S,T,R \in \Sigma^*$, $\lambda \in \mathsf{R}^+ \cup \{0\}$ We have:

$$\lambda\left[|S| + |T| - 2\sum_i min\left(f_i^{(S)}, f_i^{(T)}\right)\right] \leq$$
$$\lambda\left[|S| + |R| - 2\sum_i min\left(f_i^{(S)}, f_i^{(R)}\right)\right] + \lambda\left[|R| + |T| - 2\sum_i min\left(f_i^{(R)}, f_i^{(T)}\right)\right] \quad (5\text{-}5)$$

The proof of this lemma is presented in the Appendix. Another alternative proof can be obtained as a special case of the proof of Theorem 3 presented later in this chapter in Section 5.4.2.

Proof of Theorem 1

(p1) $EED(S,T) = EED(T,S)$ (this is obvious).

(p2) Since for all S in Σ^* we have $|S| = \sum_{a \in \Sigma} f_a^{(S)}$ we can easily verify that:

$$\lambda\left[|S| + |T| - 2\sum_{a \in \Sigma} min\left(f_a^{(S)}, f_a^{(T)}\right)\right] \geq 0 \qquad \forall S,T \in \Sigma^* \tag{5-6}$$

Let us prove first that $EED(S,T) = 0 \Rightarrow S = T$:

If $EED(S,T) = 0$, and taking into account (5-6), we get the two following relations:

$$\lambda\left[|S| + |T| - 2\sum_{a \in \Sigma} min\left(f_a^{(S)}, f_a^{(T)}\right)\right] = 0 \tag{5-7}$$

$$ED(S,T) = 0 \tag{5-8}$$

From (5-8), and since ED is metric we get: $S = T$.

The backward implication $S = T \Rightarrow EED(S,T) = 0$ is obvious.

(p3) $EED(S,T) \leq EED(S,R) + EED(R,T)$

$\forall S,T,R$ in Σ^* . Since ED is metric we can write:

$$ED(S,T) \leq ED(S,R) + ED(R,T) \tag{5-9}$$

Taking Lemma 1 and relation (5-9) into account we get :

$$EED(S,T) \leq EED(S,R) + EED(R,T)$$

From (p1), (p2), and (p3) we conclude that EED is a metric.

Example 5.3 (Revisiting Example 5.2)

Calculating EED we see that:

$$EED(S_1, S_2) = 4, \; EED(S_1, S_3) = 6$$

Which is what we expected because, according to the concept of similarity we presented earlier, S_2 is more similar to S_1 than S_3.

Example 5.4

Given; $S_1 = marwan$, $S_2 = aarwin$ (The same S_1, S_2 as in Example 5.2). S_3 is obtained by changing S_1 in the same positions, but with different characters:

$$S_3 = rarwen$$

In this example we have a particular case where ;

$$NDC(S_1, S_2) = NDC(S_1, S_3) = 6$$

But we still have

$$EED(S_1, S_2) = 4 \text{ , } EED(S_1, S_3) = 6$$

Which means that our proposed distance kept considering S_2 closer to S_1 than S_3 is, and this is what we expect since

$$\left|\{ch_a(S_1)\} \cap \{ch_a(S_2)\}\right| = 2 \text{ , } \left|\{ch_a(S_1)\} \cap \{ch_a(S_3)\}\right| = 1$$

So it is intuitive to consider S_2 closer to S_1 than S_3 is.

(For all the other characters we have:

$$\left|\{ch_x(S_1)\} \cap \{ch_x(S_2)\}\right| = 1 \text{ , } \left|\{ch_x(S_1)\} \cap \{ch_x(S_3)\}\right| = 1 \text{)}$$

Example 5.5

We will give another example that will help show the properties of our EED. Let:

$$S_1 = narwan \text{ , } S_2 = aarwnn, S_3 = aarwxn \text{ , } S_4 = xarwnn \text{ , } S_5 = xarwxn$$

The ED between S_1 and each of the other four strings is the same (which is 2). However, we will show that EED is not the same, and that it differs according to how much any two strings differ.

i- $EED(S_1, S_2) = 2$, which is the same as their ED. The two strings S_1, S_2 have the same length, the same characters, and the same frequency for each character (in fact, one string results from the other by rearranging it). These two strings have the highest

familiarity of all the other pairs (S_1 and one of S_3, S_4, S_5) so their ED is the same as their EED.

ii- $EED(S_1, S_3) = EED(S_1, S_4) = 4$, each of S_3, S_4 results from S_2 by replacing one of the characters in S_2 by another character x (in position 5 for S_3 and position 1 for S_4) . x is a character that does not exist in S_1, so adding this "unfamiliar" character makes each of these strings less similar to S_1 than S_2 is. We also see from this case that the position at which this unfamiliar character was changed did not affect EED.

iii- If we continue this process and change the characters in position 4 in S_4 or in position 1 in S_3 with that same unfamiliar character x (in both cases we obtain S_5). In both of these cases we substitute a familiar character (a in the first case and n in the second case) with an unfamiliar character x, so there is loss of similarity compared with S_3 and S_4.

By calculating EED we see that:

$$EED(S_1, S_5) = 6$$

which is what we expected.

We see that EED was not the same in the above cases, while the ED was always the same.

Example 5.6

This example concerns strings of different lengths. Let:

$S_1 = abca$, $S_2 = aabbcc$, $S_3 = adbecf$

We see that:

$$ED(S_1, S_2) = ED(S_1, S_3) = 3$$

However, by calculating their *EED* we find out that:

$$EED(S_1, S_2) = 5 , \ EED(S_1, S_3) = 7$$

Which is intuitive.

5.2.5 Complexity Analysis

The time complexity of EED is $O(m \times n)$, where m is the length of the first string and n is the length of the second one, or $O(n^2)$ if the two strings are of the same lengths. This complexity is the same as that of the edit distance.

In order to make EED scale well when applied to time series, we can find a symbolic representation method that can allow high compression of the time series (leading to drastic length reduction), with acceptable accuracy. However, our main objective was not to apply EED to time series data, but to different types of symbolic data.

5.2.6 Experimental Validation

We conducted several experiments of times series classification task. As mentioned earlier, this new distance metric is applied to data types which are represented symbolically, whether naturally or by using a certain technique of symbolic representation. We believe that bioinformatics or textual data are the ideal domains to apply EED to. However, and because our field of research is time series, we had to test EED on time series data.

We tested our distance in a classification task based on the first near-neighbor rule (*1-NN*) on the datasets available at [190]. We used leaving-one-out cross validation.

The First Experiment

The aim of this experiment is to make a direct comparison between ED,EED and SAX. We chose SAX because it is one of the most competitive dimensionality reduction techniques, but also because it is a symbolic method.

We have to mention that the datasets used in all the experiments on SAX in this dissertation were normalized because SAX can only be applied to normalized time series.

In our experiments SAX was applied as follows: the time series were represented by SAX in the manner described in Section 5.1.2. We proceeded in the same way for steps 1 and 2 to get a symbolic representation of the time series, then in step 3 we compared EED, with ED and with the distance measure defined in SAX, on the resulting strings.

For this experiment, we used the same compression ratio that was used to test SAX (i.e. 1 to 4). We also used the same range of alphabet size (3:10).

For each dataset we tune the parameters on the training set to get the optimal values of these parameters; i.e. the values that minimize the error. Then we use these optimal

values on the testing set to get the error rate for each method and for each dataset. As for parameter λ , for simplicity, and in all the experiments we conducted, we optimized it in the interval [0, 1] only (step=0.25), except in the cases where there was strong evidence that the error was decreasing monotonously as λ increased.

For this experiment, we chose, at random, 4 datasets from the 20 datasets of [190]. The chosen datasets were (*CBF)*, (*Trace*), (*Two_Patterns*), (*Yoga*). After optimizing the parameters on the training sets, we used these parameters on the testing sets of these datasets. We got the results shown in Table. 5.1 (The best results are boldfaced and shaded)

Table 5.1. The error rate of ED, EED, SAX on the testing sets of (*CBF*), (*Trace*), (*Two Patterns*), and *(Yoga*). The parameters used in the calculations are those that give optimal results on the training sets, the alphabet size was chosen from the interval (3:10).The compression ratio is 1 to 4

	The Edit Distance (ED)	**The Extended Edit Distance (EED)**	**SAX**
CBF	0.029 α* =10	**0.026** α =3, λ =0.75	0.104 α =10
Trace	**0.11** **α =10**	**0.07** **α =6, λ ≥ 1.25**	**0.42** **α =10**
Two_Patterns	**0.015** **α =3**	**0.015** **α =3, λ =0**	**0.081** **α =10**
Yoga	**0.155** **α =7**	**0.155** **α =7, λ =0**	**0.199** **α =10**

(*: α is the alphabet size)

The results show that EED was always better, or equal, to the other methods.

The Second Experiment

This experiment is an extension of the first experiment; we did not compare our new distance with ED and SAX only, but we also compared it with other distances that are applied to non-compressed time series. We chose the two most famous distances: DTW and the Euclidean distance. We chose randomly 4 datasets of the remaining datasets in [190]. These were (*Gun_Point*), (*OSU Leaf*), (*50words*), and (*Fish*). We used the same compression ratio and the same range of alphabet size that we used the first experiment. We proceeded in the same way. The results that we obtained are shown in Table 5.2.

Table 5.2. The error rate of ED, EED, SAX, DTW together with the Euclidean distance on the testing sets of (*Gun_Point*), (*OSU Leaf*), (*50words*), and (*Fish*). The alphabet size was chosen from the interval (3:10). The compression ratio is 1 to 4

	Euclidean Distance	DTW	ED	EED	SAX
Gun-Point	0.087	0.093	0.073 α =4	**0.06** α =4, λ =0.25	0.233 α =10
OSULeaf	0.483	0.409	0.318 α =5	**0.293** α =5, λ =0.75	0.475 α =9
50words	0.369	0.310	**0.266** α =7	**0.266** α =7, λ =0	0.327 α =9
Fish	0.217	0.267	**0.149** α =10	**0.149** α =10, λ =0	0.514 α =10

The results of this experiment show that EED is superior to the other distances.

The Third Experiment

This experiment aims for studying the impact of using a wider range of alphabet size: (3:20). We proceed in the same way we did before; we randomly chose 6 datasets of the remaining datasets. The 6 chosen datasets were (*Coffee*), (*Beef*), (*Adiac*), (*ECG200*), (*Wafer*), and (*Face all*). The compression ratio is the same as before (1 to 4). *EED* was compared with ED and SAX. The final results on the testing sets are shown in Table 5.3.

Table 5.3. The error rate of ED, EED, SAX on the testing sets of *(Coffee*), (*Beef*), (*Adiac*), (*ECG200*), (*Wafer*), and (*Face all*)**.** The alphabet size was chosen from the interval (3:20).The compression ratio is (1 to 4)

	The Edit Distance (ED)	The Extended Edit Distance (EED)	SAX
Coffee	0.071 α =12,13	**0.0** α =14, λ =0.25	0.143 α =20
Beef	0.467 α =17	**0.4** α =4, λ= 0.75	0.433 α =20
Adiac	0.555 α =18	**0.524** α =19, λ =1	0.867 α =18
ECG200	0.23 α =13	0.19 α =5, λ =0.25	**0.13** α =16
Wafer	0.008 α =4	0.008 α =4, λ =0	**0.004** α =19
Face (all)	0.324 α =7	0.324 α =7, λ =0	**0.305** α =19

The results of this experiment show that EED is the best one.

The Fourth Experiment

This experiment is designated to study the impact of using a different compression ratio. This means that in this case we proceeded in the same way as in step 1 in Section 5.2.1, but later we used a different PAA representation than the one we used in the previous experiments. We conducted this experiment on the rest of the datasets in [190].The compression ratio of this experiment is (1 to 5). The alphabet range is (3:10). After proceeding in the same way that we used for the other experiments we got the results shown in Table 5.4

Table 5.4. The error rate of ED, EED, SAX on the testing sets of *(Lighting2*), (*Lighting7*), (*Synthetic Control*), (*Face four),*(*Trace*), and (*Olive Oil*) the alphabet size was chosen from the interval (3:10).The compression ratio is (1 to 5)

	The Edit Distance (ED)	**The Extended Edit Distance (EED)**	**SAX**
Lighting2	**0.311** α =5	**0.311** α =5, λ =0,0.75	0.377 α =3
Lighting7	**0.247** **α =5**	**0.247** **α =5, λ= 0**	**0.479** **α =7**
Trace	**0.11** **α =10**	**0.09** **α =8, λ =0.75**	**0.36** **α =10**
Synthetic Control	**0.077** **α =8**	**0.05** **α =6, λ =0.25**	**0.03** **α =10**
Face (four)	**0.045** **α =5,6**	**0.045** **α =5,6, λ =0**	**0.182** **α =9**
Olive Oil	**0.267** **α =7**	**0.267** **α =7, λ =0,...,1**	**0.833** $\forall \alpha$

The results obtained show that EED is the best one.

Remark :

When evaluating the performance of two methods, the classification error should not be considered independently of the number of classes. For instance, the classification error of SAX on both Adiac (0.867, 37 classes, Table 5.3) and OliveOil (0.833, 4 classes, Table 5.3) is almost the same. However, the performance of the method on Adiac is actually better than that on OliveOil because with the first data set the error of using a random classifier is 36/37=0.973, so the method did outperform a random classifier, while in the case of OliveOil the error of using a random classifier is 3/4 =0.750, which is lower than the error of applying the method.

5.2.7 Other Applications

As mentioned earlier, when we thought of EED we wanted to propose a distance that could be applied to data types that are naturally symbolic, or represented symbolically by a representation technique, but we could not test our distance on any type of data other than symbolically represented time series because this is our field of research. But after publishing our work, a new work was published [14] in which the author applied our proposed distance metric EED to evaluate the performance of range queries in the *Recursive Lists of Clusters* (RLC) metric data structure, when the metric spaces are natural language dictionaries on which EED is defined. Her experiments show that RLC has a good performance in all the tested cases when using EED as a similarity distance.

5.2.8 Discussion

In this section we presented a new distance metric applied to strings. The main feature of this distance is that it considers the frequency of characters, which is something other distance measures do not consider. Another important feature of this distance is that it is metric. We tested this new distance on a time series classification task, and we compared it to other distances. We showed that our distance gave better results in most cases.

We think that the main drawback of this distance is that it uses the parameter λ, which is heuristic and it also increases the training phase.

In the experiments we conducted we had to use time series of equal lengths for comparison reasons only, since SAX can be applied only to strings of equal lengths. But EED (and ED, too) can be applied to strings of different lengths.

Although there are other similarity measures and distance metrics that are derived from the edit distance and which are applied to time series data mining and information retrieval, we did not compare our method with any of them because we view EED as a distance that can be applied to different symbolic data types, while those distances and similarity measures were developed only for the field of time series data mining and information retrieval.

We did not conduct experiments for alphabet size=2 because SAX is not applicable in this case (when alphabet size =2 then the distance between any two strings will be zero with SAX, and for any dataset). However, it is important to mention that comparing EED or ED, with SAX was only used as an indicator of performance. In fact, SAX is faster than any of EED or ED, even though the error it produces is greater, or even much greater, in most cases than that of EED or ED.

In order to represent the time series symbolically, we had to use a technique prepared for SAX for comparison purposes. Nevertheless, a representation technique prepared specifically for EED may even give better results.

The main property of EED over ED is that it is more precise, since it considers a global level of similarity that ED does not consider.

In order to interpret the results correctly, we have to remember that the comparison was based on a time series classification task. However, we did not develop EED, or the other extensions of the edit distance, for this purpose, so we think the performance of our extensions would be better on other time series data mining tasks.

5.3 The Multi-resolution Extended Edit Distance (MREED)

Distance metrics and similarity measures that we know of compare two strings on a symbol-to-symbol basis, which reduces the number of possible applications of the symbolic representation. In this section we move one step forward by comparing subsequences. In this case, we propose a modification of the edit distance which considers the frequencies of substrings as well as the frequencies of single characters.

5.3.1 Definition-MREED

Let Σ be a finite alphabet, and let Σ^* be the set of strings on Σ, and let $f_i^{(S)}$, $f_i^{(T)}$ be the frequency of the character i in S and T, respectively. $ff_{ij}^{(S)}$, $ff_{ij}^{(T)}$ be the frequency of the two-character subsequence ij in S and T, respectively (including the case where $i = j$), and where S, T are two non-empty strings on Σ.

The *Multi-Resolution Extended Edit Distance* (MREED) is defined as:

$$MREED\ (S,T) = ED(S,T) + \lambda\left[\,|S| + |T| - 2\sum_i min(f_i^{(S)}, f_i^{(T)})\right]$$
$$+ \delta\left[\,|S| + |T| - 2(\sum_i \sum_j min(ff_{ij}^{(S)}, ff_{ij}^{(T)}) + 1)\right] \qquad (5\text{-}10)$$

Where $|S|$, $|T|$ are the lengths of the strings S, T respectively and where $\lambda \geq 0$, $\delta \geq 0$ ($\lambda, \delta \in \mathbb{R}$). We call λ the frequency factor of the first degree, and δ the frequency factor of the second degree.

5.3.2 Theorem 2

MREED is a metric distance □

Proof of Theorem 2

We can easily notice that:

$$\delta\left[|S|+|T|-2\left(\sum_i\sum_j min\left(ff_{ij}^{(S)}, ff_{ij}^{(T)}\right)+1\right)\right] \geq 0 \qquad \forall S,T \in \Sigma^* \quad (5\text{-}11)$$

The rest of the proof is similar to that of Theorem 1.

Example 5.7

Given the following strings:

$S_1 = abca,\ S_2 = aabbcc\ , S_3 = adbecf$.

Intuitively, we see that S_2 is closer to S_1 than S_3 .Yet, if we calculate their edit distance we see that:

$$ED(S_1, S_2) = ED(S_1, S_3) = 3\ .$$

But if we apply MREED we see that:

$$MREED(S_1, S_2) = 9\ ,\ MREED(S_1, S_3) = 15\ .$$

So we see that our distance could detect that S_2 is closer to S_1 than S_3

5.3.3 Complexity Analysis

The time complexity of MREED is $O(m \times n)$, which is the same as that of ED.

5.3.4 Experimental Evaluation

The experimental protocol that we followed to test MREED is similar to that we used to test EED. We tested our distance on 12 datasets chosen from the 20 datasets available at [190]. We meant to include quite a variety of cases in our tests; the number of classes varies between 2 (*Gun-Point*) and 50 (*50words*). The size of the training set varies between 30 (*Beef* and *CBF*) and 560 (*Face all*). The size of the testing set varies between 30 (*Beef* and *Olive Oil*) and 3000 (*Yoga*), and the length of the time series (before compression) varies between 60 (*Synthetic Control*) and 570 (*Olive Oil*).

We compared MREED with ED, and SAX using the same alphabet size and compression ratio that SAX uses. We also compared our distance with the Euclidean distance on a *1-NN* classification task.

Both ED and SAX have one parameter, which is the alphabet size. MREED has two extra parameters: the frequency factor of the first degree λ , and the frequency factor of the second degree δ . For each of the 12 datasets we started by tuning the two parameters λ and δ on the training sets to get the optimal values of these parameters; i.e. the values that minimize the error rate. For simplicity, we optimized parameters λ and δ in the interval $[0,1]$ only (step=0.25), except in the cases where there was strong evidence that the error is decreasing monotonously as λ or δ increases.

After optimizing the two parameters on the training sets for the 12 datasets we got the results shown in Table 5.5 (There is no training phase for the Euclidian distance).

Table 5.5. A comparison of ED, MREED, SAX on the training sets of 12 different datasets. The table shows the error rates for each distance and for each dataset.

	ED	MREED	SAX
Synthetic Control	0.037	0.033	**0.027**
Gun-Point	**0.02**	**0.02**	0.08
CBF	0.033	**0**	0.167
Face (all)	0.157	0.157	**0.118**
OSULeaf	0.2	**0.19**	0.365
SwedishLeaf	0.34	**0.316**	0.486
50words	**0.253**	**0.253**	0.349
Trace	0.05	**0**	0.31
Adiac	0.687	**0.674**	0.918
Yoga	**0.193**	**0.193**	0.24
Beef	0.533	**0.433**	0.467
OliveOil	0.333	**0.3**	0.833

Then we used the optimal parameters for each dataset, and for each method, on the testing sets, we obtained the results shown in Table 5.6 (the cases where the Euclidean distance outperformed the other methods were shown in boldfaced red)

Table 5.6 The error rates of ED, MREED, SAX , together with the Euclidean distance on the testing sets of the 12 datasets. The parameters used in the calculations are those that give optimal results on the training sets

	1-NN Euclidean distance	**ED**	**MREED**	**SAX**
Synthetic Control	0.12	0.037 α =7	0.053 α =8, λ=0, δ=0.25	**0.033** α =10
Gun-Point	0.087	0.073 α =4	**0.06** α =4, λ =0.25, δ=0	0.233 α =10
CBF	0.148	0.029 α =10	**0.023** α =3, λ =0.25,0.5, δ=0.25	0.104 α =10
Face (all)	0.286	0.324 α =7	0.324 α =7=, λ =0, δ=0	**0.319** α =10
OSULeaf	0.483	0.318 α =5	**0.302** α =5, λ =0, δ=0.25	0.475 α =9
SwedishLeaf	**0.213**	**0.344** α =7	0.365 α =7, λ =0.25, δ=0	0.490 α =10
50words	0.369	**0.266** α =7	**0.266** α =7, λ =0, δ=0	0.327 α =9
Trace	0.24	0.11 α =10	**0.02** α =6, (λ =0, δ≥0.75), (λ =0,0.25, δ=1)	0.42 α =10
Adiac	0.389	0.701 α =7	**0.642** α =9, λ =0.5, δ=0	0.903 α =10
Yoga	0.170	**0.155** α =7	**0.155** α =7, λ =0, δ=0	0.199 α =10
Beef	0.467	0.467 α =4	**0.367** α =4, λ =0.5, δ=0.25	0.533 α =10
OliveOil	**0.133**	0.467 α =9	**0.367** α =9,(λ=0.75,δ≥0.5), (λ =1, δ≥0.75)	0.833 ∀ α

The results show that the performance of MREED it better than that of both SAX, and ED.

It is worth mentioning that for the Euclidian distance there is no compression of data, so in some cases it may give better results than symbolic, compressed distances.

5.3.5 Discussion

In this section we presented another extension of the edit distance which is MREED . The main feature of this distance is that it does not only consider the frequencies of single characters but also the frequencies of sub-sequences. We tested this distance on a time series classification task, and we compared it to two other distances. We showed that our distance gave better results even when compared to a method (SAX) that is designed mainly for symbolically represented time series.

5.4 The Σ_{GRAM} Distance

Counting the occurrences of a pattern in a string was first introduced by [192] where the author used this concept to address the approximate string matching problem. The author presented a similarity measure based on this concept. However, the model he proposed violates the metric axioms, so it could only produce approximate solutions.

In this section we establish a general metric model of the distances we presented in Sections 5.2 and 5.3. This distance that we call the *Σ_{GRAM} Distance* is based on the frequencies of *n-grams*.

We present first a generalization of the definition we presented in Section 5.2.1

Definition-The Number of Distinct n-Grams (NDnG*)*: Given two strings S , T . The number of distinct *n-grams* (substrings of length *n*) that the two strings *S* and *T* contain is defined as:

$$ND_nG(S,T) = \left|\{n-gram(S)\} \cup \{n-gram(T)\}\right| \qquad (5\text{-}12)$$

where $n-gram(\ )$ is the set of *n-grams* that a string consists of.

Notice that : $1 \le n \le min(|S|,|T|)$

Example 5.8

Given the following strings:

$S = exogen$, $R = oxygen$, $T = emolen$

The sets of n-grams for these strings are given by:

n	***R***	***S***	***T***
1	*o, x, y, g, e, n*	*e, x, o, g, e, n*	*e, m, o, l, e, n*
2	*ox, xy, yg, ge, en*	*ex, xo, og, ge, en*	*em, mo, ol, le, en*
3	*oxy, xyg, yge, gen*	*exo, xog, oge, gen*	*emo, mol, ole, len*
4	*oxyg, xyge, ygen*	*exog, xoge, ogen*	*emol, mole, olen*
5	*oxyge, xygen*	*exoge, xogen*	*emole, molen*
6	*oxygen*	*exogen*	*emolen*

Comparing $ND_nG(S,R)$, and $ND_nG(S,T)$ gives:

n	***NDnG(S,R)***	***NDnG(S,T)***
1	5	4
2	2	1
3	1	0
4	0	0
5	0	0
6	0	0
$\sum_{n=1}^{6} ND_nG$	8	5

The above comparison shows a greater similarity between S and R than between S and T, which is intuitive. But if we compute the edit distance we get:

$$ED(S,R) = ED(S,T) = 2$$

5.4.1 Definition-The Σ_{GRAM} Distance

Let Σ be a finite alphabet, and let Σ^* be the set of strings on Σ. An *n-gram* is a substring of *n* characters from a given string.

Given *n*, Let $f_{a_n}^{(S)}$ be the frequency of the *n-gram* a_n in S, and $f_{a_n}^{(T)}$ be the frequency of the *n-gram* a_n in T, where S, T are two strings in Σ^*. Let $\mathbb{N}$ be the set of integers, and $\mathbb{N}^+$ the set of positive integers.

For notation convenience, we define the function:

$$g : \mathbb{N}^+ \times \Sigma^* \to \mathbb{N}$$

$$g(n,S) = n \qquad \text{if} \quad 1 \le n \le |S|$$
$$g(n,S) = |S| + 1 \qquad \text{if} \quad |S| < n$$

The $\sum_{GRAM}$ distance between *S* and *T* is thus defined as:

$$\sum_{GRAM}(S,T) = \sum_{n=1}^{max(|S|,|T|)} \lambda_n \cdot \left[|S| + |T| - g(n,S) - g(n,T) + 2 - 2 \cdot \sum_{a_n \in A^n} min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)} \right) \right]$$

(5-13)

where $|S|, |T|$ are the lengths of the two strings S, T respectively, and where $\lambda_n \in \mathbb{R}^+ \cup \{0\}$ □

Since $|S| = n - 1 + \sum_{a_n \in \Sigma^n} f_{a_n}^{(S)} \quad \forall \quad 1 \le n \le |S|$, and $\sum_{a_n \in \Sigma^n} f_{a_n}^{(S)} = 0$ for all $n > |S|$, we can write:

$$|S| = g(n,S) - 1 + \sum_{a_n \in \Sigma^n} f_{a_n}^{(S)} \quad \forall \quad n \in \mathbb{N}^+ .$$

Thus for $n > |S|$ we have :

$$|S| - g(n,S) + 1 = 0$$

and $$\sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right) = 0$$

And for $n > |T|$ we have :

$$|T| - g(n,T) + 1 = 0$$

and $$\sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right) = 0$$

Consequently, definition (5-13) can be written as:

$$\sum\nolimits_{GRAM}(S,T) =$$
$$\sum_{n=1}^{\infty} \lambda_n \cdot \left[|S| + |T| - g(n,S) - g(n,T) + 2 - 2 \cdot \sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right) \right]$$

or :

$$\sum\nolimits_{GRAM}(S,T) = \sum_{n=1}^{\infty} \lambda_n . D_n(S,T) \qquad (5\text{-}14)$$

where

$$D_n(S,T) = \sum_{a_n \in \Sigma^n} f_{a_n}^{(S)} + \sum_{a_n \in \Sigma^n} f_{a_n}^{(T)} - 2 \cdot \sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right)$$

5.4.2 Theorem 3

Σ_{GRAM} is a metric distance □

Proof of Theorem 3

(p1) $\sum_{GRAM}(S,T) = \sum_{GRAM}(T,S)$ (obvious).

(p2) It is easy to notice that $D_n \geq 0 \quad \forall n \in \mathbb{N}^+$

$$\Rightarrow \sum_{GRAM}(S,T) = \sum_{n=1}^{\infty} D_n(S,T) \geq 0 \qquad \forall S,T \in \Sigma^*$$

i- Let us prove first that: $\sum_{GRAM}(S,T) = 0 \Rightarrow S = T$.

If $\sum_{GRAM}(S,T) = 0$, and taking into account (5-14) then each term D_n of the summation in equation (5-13) should be equal to 0.

In particular for $n = M = max(|S|,|T|)$ we have:

$$D_M(S,T) = \sum_{a_M \in \Sigma^M} f_{a_M}^{(S)} + \sum_{a_M \in \Sigma^M} f_{a_M}^{(T)} - 2 \cdot \sum_{a_M \in \Sigma^M} min\left(f_{a_M}^{(S)}, f_{a_M}^{(T)}\right) = 0 \quad (5\text{-}15)$$

If $|S| < M \Rightarrow$

$$\sum_{a_M \in \Sigma^M} f_{a_M}^{(S)} = 0, \quad \sum_{a_M \in \Sigma^M} f_{a_M}^{(T)} = 1 \quad \text{and} \quad \sum_{a_M \in \Sigma^M} min\left(f_{a_M}^{(S)}, f_{a_M}^{((T)}\right) = 0$$

Which contradicts (5-15).

The same applies if $|T| < M$

This implies that $|S| = |T| = M$ and $\sum_{a_M \in \Sigma^M} min\left(f_{a_M}^{(S)}, f_{a_M}^{((T)}\right) = 1$ which implies that $S = T$.

ii- $S = T \Rightarrow \sum_{GRAM}(S,T) = 0$ (obvious)

From i and ii we get:

$$\sum_{GRAM}(S,T) = 0 \Leftrightarrow S = T$$

(p3) The triangle inequality

Let : $$D_n(S,T) = |S| + |T| - 2\cdot(n-1) - 2\cdot \sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right).$$

We will show that:

$$D_n(S,T) \le D_n(S,R) + D_n(R,T) \qquad \forall S,T,R \in \Sigma^* \tag{5-16}$$

First, we notice that the following equivalences hold:

$$D_n(S,T) \le D_n(S,R) + D_n(R,T) \Leftrightarrow$$

$$|S| + |T| - 2\cdot(n-1) - 2\cdot \sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right) \le$$

$$|S| + |R| - 2\cdot(n-1) - 2\cdot \sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(S)}, f_{a_n}^{(R)}\right) + |R| + |T| - 2\cdot(n-1) - 2\cdot \sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(R)}, f_{a_n}^{(T)}\right)$$

$$\Leftrightarrow \sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(S)}, f_{a_n}^{(R)}\right) + \sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(R)}, f_{a_n}^{(T)}\right) \le |R| - (n-1) + \sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right)$$

Since $|R| = n - 1 + \sum_{a_n \in \Sigma^n} f_{a_n}^{(R)}$ this implies that proving (5-16) is equivalent to proving the following:

$$\sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right) + \sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(R)}, f_{a_n}^{(T)}\right) \le \sum_{a_n \in \Sigma^n} f_{a_n}^{(R)} + \sum_{a_n \in \Sigma^n} min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right) \tag{5-17}$$

Second, we notice that:

$$min\left(f_{a_n}^{(S)}, f_{a_n}^{(R)}\right) \le f_{a_n}^{(R)}\ , min\left(f_{a_n}^{(T)}, f_{a_n}^{(R)}\right) \le f_{a_n}^{(R)} \qquad \forall a_n \in \Sigma^n$$

In addition, $\forall a_n \in \Sigma^n$ we have:

- If $f_{a_n}^{(R)} = min\left(f_{a_n}^{(R)}, f_{a_n}^{(S)}, f_{a_n}^{(T)}\right) \Rightarrow$

$$min\left(f_{a_n}^{(R)}, f_{a_n}^{(T)}\right) \le min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right) \Rightarrow$$
$$min\left(f_{a_n}^{(S)}, f_{a_n}^{(R)}\right) + min\left(f_{a_n}^{(R)}, f_{a_n}^{(T)}\right) \le f_{a_n}^{(R)} + min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right)$$

- If $f_{a_n}^{(S)} = min\left(f_{a_n}^{(R)}, f_{a_n}^{(S)}, f_{a_n}^{(T)}\right) \Rightarrow$
  $min\left(f_{a_n}^{(S)}, f_{a_n}^{(R)}\right) \le min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right) \Rightarrow$
  $min\left(f_{a_n}^{(S)}, f_{a_n}^{(R)}\right) + min\left(f_{a_n}^{(R)}, f_{a_n}^{(T)}\right) \le f_{a_n}^{(R)} + min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right)$

- If $f_{a_n}^{(T)} = min\left(f_{a_n}^{(R)}, f_{a_n}^{(S)}, f_{a_n}^{(T)}\right) \Rightarrow$
  $min\left(f_{a_n}^{(R)}, f_{a_n}^{(T)}\right) \le min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right) \Rightarrow$
  $min\left(f_{a_n}^{(S)}, f_{a_n}^{(R)}\right) + min\left(f_{a_n}^{(R)}, f_{a_n}^{(T)}\right) \le f_{a_n}^{(R)} + min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right)$

This means that $\forall a_n \in \Sigma^n$ we have:

$$min\left(f_{a_n}^{(S)}, f_{a_n}^{(R)}\right) + min\left(f_{a_n}^{(R)}, f_{a_n}^{(T)}\right) \le f_{a_n}^{(R)} + min\left(f_{a_n}^{(S)}, f_{a_n}^{(T)}\right)$$

Summing over all a_n in Σ^n we get the proof of proposition (5-17) thus the proof of (5-16).

Summing over n we get the proof of (**p3**).

From (**p1**), (**p2**), and (**p3**) we conclude that Σ_{GRAM} is a metric distance.

5.4.3 Discussion

In this section we extended the edit distance to consider the frequencies of characters and also all substrings of a given string. The main feature of this model is that it is based not only on the simple model of symbolic distances which compare a symbol to another symbol, but it permits comparing substrings as well.

A possible future application is to use this model in motif discovery in time series data mining. The basis of this application is to represent the motif symbolically and apply our model to consider the frequency of the motif.

5.5 The Parameter-free Extended Edit Distance (PFEED)

Despite all the advantages that all the extensions of the edit distance we presented in the previous section have, they have one particular inconvenience; they all contain parameters in their definition. These parameters are not semantic. They also require tuning on the training sets to get their optimal value, and this training phase can be long. Although there are alternatives to the protocol we followed in our experiments, which can speed up the training phase, we thought that a non-parametric extension to the edit distance can be beneficial, especially in the cases where there is no possibility of a training phase.

In this section we present another extension to the edit distance which we call the *Parameter-free Extended Edit Distance* (PFEED). As indicated in the introduction of this dissertation and of this chapter, the in which we presented this distance has been accepted for publication but has not been published, so this section is presented for the first time here.

5.5.1 Definition-The Parameter-free Extended Edit Distance (PFEED)

Let $\sum$ be a finite alphabet, and let $f_i^{(S)}$, $f_i^{(T)}$ be the frequencies of the character i in S and T, respectively. The *Parameter-free Extended Edit Distance* (PFEED) is defined as:

$$PFEED\left(S,T\right)=ED\left(S,T\right)+\left(1-\frac{2\sum_{i}min\left(f_i^{(S)},f_i^{(T)}\right)}{\left|S\right|+\left|T\right|}\right) \tag{5-18}$$

Where $\left|S\right|,\left|T\right|$ are the lengths of the two strings S,T respectively

5.5.2 Theorem 4

PFEED is a metric distance □

Before we prove the theorem we introduce the following lemma:

Lemma 2

Let $\sum$ be a finite alphabet, $f_i^{(S)}$ be the frequency of the character i in S, where S is a string on $\sum$. Then $\forall S_1, S_2, S_3$ we have:

$$\left(1 - \frac{2\sum_i min\left(f_i^{(S_1)}, f_i^{(S_2)}\right)}{|S_1| + |S_2|}\right) \leq \left(1 - \frac{2\sum_i min\left(f_i^{(S_1)}, f_i^{(S_3)}\right)}{|S_1| + |S_3|}\right) + \left(1 - \frac{2\sum_i min\left(f_i^{(S_3)}, f_i^{(S_2)}\right)}{|S_3| + |S_2|}\right) \quad (5\text{-}19)$$

$\forall n$, where n is the number of characters used to represent the strings

Proof

i) $n = 1$, this is a trivial case, where the strings are represented with one character .

Given three strings S_1, S_2, S_3 represented by the same character a .

Let $f_a^{(S_1)}, f_a^{(S_2)}, f_a^{(S_3)}$ be the frequency of a in S_1, S_2, S_3, respectively. We have six configurations in this case:

1) $f_a^{(S_1)} \leq f_a^{(S_2)} \leq f_a^{(S_3)}$
2) $f_a^{(S_1)} \leq f_a^{(S_3)} \leq f_a^{(S_2)}$
3) $f_a^{(S_2)} \leq f_a^{(S_1)} \leq f_a^{(S_3)}$
4) $f_a^{(S_2)} \leq f_a^{(S_3)} \leq f_a^{(S_1)}$
5) $f_a^{(S_3)} \leq f_a^{(S_1)} \leq f_a^{(S_2)}$
6) $f_a^{(S_3)} \leq f_a^{(S_2)} \leq f_a^{(S_1)}$

We will prove that relation (5-19) holds in these six configurations.

1) $f_a^{(S_1)} \leq f_a^{(S_2)} \leq f_a^{(S_3)}$

In this case we have:

$$min\left(f_a^{(S_1)}, f_a^{(S_2)}\right) = f_a^{(S_1)}, min\left(f_a^{(S_1)}, f_a^{(S_3)}\right) = f_a^{(S_1)}, min\left(f_a^{(S_2)}, f_a^{(S_3)}\right) = f_a^{(S_2)}$$

By substituting the above values in (5-19) we get:

$$1 - \frac{2\,min\left(f_a^{(S_1)}, f_a^{(S_2)}\right)}{f_a^{(S_1)} + f_a^{(S_2)}} \le 1 - \frac{2\,min\left(f_a^{(S_1)}, f_a^{(S_3)}\right)}{f_a^{(S_1)} + f_a^{(S_3)}} + 1 - \frac{2\,min\left(f_a^{(S_3)}, f_a^{(S_2)}\right)}{f_a^{(S_3)} + f_a^{(S_2)}}$$

$$1 - \frac{2\,f_a^{(S_1)}}{f_a^{(S_1)} + f_a^{(S_2)}} \le 1 - \frac{2\,f_a^{(S_1)}}{f_a^{(S_1)} + f_a^{(S_3)}} + 1 - \frac{2\,f_a^{(S_2)}}{f_a^{(S_3)} + f_a^{(S_2)}}$$

$$\frac{2\,f_a^{(S_1)}}{f_a^{(S_1)} + f_a^{(S_2)}} \ge \frac{2\,f_a^{(S_1)}}{f_a^{(S_1)} + f_a^{(S_3)}} + \frac{2\,f_a^{(S_2)}}{f_a^{(S_3)} + f_a^{(S_2)}} - 1$$

If we substitute $f_a^{(S_2)}, f_a^{(S_1)}, f_a^{(S_2)}$ with $f_a^{(S_1)}, f_a^{(S_3)}, f_a^{(S_3)}$, respectively in the denominators of the last relation it still holds according to the stipulation of this configuration. We get:

$$\frac{2\,f_a^{(S_1)}}{f_a^{(S_1)} + f_a^{(S_1)}} \ge \frac{2\,f_a^{(S_1)}}{f_a^{(S_3)} + f_a^{(S_3)}} + \frac{2\,f_a^{(S_2)}}{f_a^{(S_3)} + f_a^{(S_3)}} - 1$$

$$\frac{2\,f_a^{(S_1)}}{2\,f_a^{(S_1)}} \ge \frac{2\,f_a^{(S_1)}}{2\,f_a^{(S_3)}} + \frac{2\,f_a^{(S_2)}}{2\,f_a^{(S_3)}} - 1$$

$$1 \ge \frac{f_a^{(S_1)} + f_a^{(S_2)}}{f_a^{(S_3)}} - 1$$

$$2\,f_a^{(S_3)} \ge f_a^{(S_1)} + f_a^{(S_2)}$$

This is valid according to the stipulation of this configuration.

The proofs of cases 2), 3), 4), 5) and 6) are similar to that of case 1).

From 1)-6) we conclude that the lemma is valid for $n = 1$

ii) $n > 1$

This is a generalization of the case where $n = 1$.

$\forall i \in n$, then

$$\left(1-\frac{2\,min\left(f_i^{(S_1)},f_i^{(S_2)}\right)}{|S_1|+|S_2|}\right)\le\left(1-\frac{2\,min\left(f_i^{(S_1)},f_i^{(S_3)}\right)}{|S_1|+|S_3|}\right)+\left(1-\frac{2\,min\left(f_i^{(S_3)},f_i^{(S_2)}\right)}{|S_3|+|S_2|}\right)$$

holds, according to the first case i)

By summing over n we get

$$\left(1-\frac{2\sum_i min\left(f_i^{(S_1)},f_i^{(S_2)}\right)}{|S_1|+|S_2|}\right)\le\left(1-\frac{2\sum_i min\left(f_i^{(S_1)},f_i^{(S_3)}\right)}{|S_1|+|S_3|}\right)+\left(1-\frac{2\sum_i min\left(f_i^{(S_3)},f_i^{(S_2)}\right)}{|S_3|+|S_2|}\right)$$

Proof of Theorem 4

Before we prove the theorem, we can easily notice that:

$$\left(1-\frac{2\sum_i min\left(f_i^{(S)},f_i^{(T)}\right)}{|S|+|T|}\right)\ge 0 \qquad \forall S,T \tag{5-20}$$

In order to prove the theorem we have to prove that:

(**p1**) $PFEED(S,T)=0 \Leftrightarrow S=T$

a) $PFEED(S,T)=0 \Rightarrow S=T$

If $PFEED(S,T)=0$, and taking into account (5-20), we get the following relation:

$$\left(1-\frac{2\sum_i min\left(f_i^{(S)},f_i^{(T)}\right)}{|S|+|T|}\right)=0 \tag{5-21}$$

$$ED(S,T)=0 \tag{5-22}$$

From (5-22), and since ED is metric we get: $S=T$

b) $S = T \Rightarrow PFEED(S,T) = 0$ (obvious).

From **a**) and **b**) we get $PFEED(S,T) = 0 \Leftrightarrow S = T$

(**p2**) $PFEED(S,T) = PFEED(T,S)$ (obvious).

(**p3**) $PFEED(S,T) \le PFEED(S,R) + PEEED(R,T)$

$\forall S, T, R$, we have:

$$ED(S,T) \le ED(S,R) + ED(R,T) \tag{5-23}$$

(Valid since ED is metric)

From Lemma 2 we have:

$$\left(1 - \frac{2\sum_i \min(f_i^{(S)}, f_i^{(T)})}{|S|+|T|}\right) \le \left(1 - \frac{2\sum_i \min(f_i^{(S)}, f_i^{(R)})}{|S|+|R|}\right) + \left(1 - \frac{2\sum_i \min(f_i^{(R)}, f_i^{(T)})}{|R|+|T|}\right) \tag{5-24}$$

Adding (5-23), (5-24) side to side we get:

$$PFEED(S,T) \le PFEED(S,R) + PFEED(R,T)$$

5.5.3 Experiments

We chose at random 12 datasets of the 20 data sets at [190] using the same protocol we described in Section 5.2.6. We compared PFEED with ED on a *1-NN* classification task. The compression ratio was 1 to 4. The alphabet size ranged in the interval (3:10). Although the proposed distance does not include any parameters, the experiments included a training phase to choose the optimal value of the alphabet size, which is a parameter related to the symbolic representation of SAX and not to PFEED. Table 5.7 shows the results we obtained.

Table 5.7 The error rates of ED, PFEED on 12 datasets.

	ED	PFEED
Synthetic Control	0.037 α =7	**0.03** α =7
Gun-Point	**0.073** **α =4**	**0.067** **α =4**
CBF	**0.029** **α =10**	**0.01** **α =6**
Face (all)	**0.324** **α =7**	**0.323** **α =5**
OSULeaf	**0.318** **α =5**	**0.310** **α =5**
50words	**0.266** **α =7**	**0.270** **α =7**
Trace	**0.11** **α =10**	**0.09** **α =8,10**
Lighting 7	**0.247** **α =10**	**0.232** **α =10**
Adiac	**0.701** **α =7**	**0.650** **α =9**
Yoga	**0.155** **α =7**	**0.156** **α =8**
Beef	**0.467** **α =4**	**0.433** **α =4,6**
Coffee	**0.107** **α =8**	**0.071** **α =8**

The results show that PFEED gives better results than ED.

We also compared PFEED with MREED, one of the parametric extensions of the edit distance and which we presented in Section 5.3, using 8 datasets chosen randomly from the datasets presented in 5.5.3 and for the same range on alphabet size. Table 5.8 shows the results of this experiment.

The results obtained show that the performance of this non-parametric extension is acceptable compared with that of MREED which uses two parameters (λ,δ) and requires a training phase.

Table 5.8 The error rates of PFEED, MREED on 8 of the 12 datasets presented in Table 5.7

	PFEED	MREED
Synthetic Control	**0.03**	0.053
Gun-Point	0.067	**0.06**
CBF	**0.01**	0.023
Face (all)	**0.323**	0.324
OSULeaf	0.310	**0.302**
50words	0.270	**0.266**
Trace	0.09	**0.02**
Adiac	0.650	**0.642**
Yoga	0.156	**0.155**
Beef	0.433	**0.367**

5.5.4 Discussion

In this section we presented a non-parametric extension of the edit distance which is metric. The main advantage of this version compared with the other extensions is that its training time is much shorter.

Again we have to emphasize on the fact that the main objective of developing this extension was to apply it to other data types where we are expecting to get even better results.

5.6 Enhancing SAX Using Updated Lookup Tables

In this section we present the second main contribution of this chapter which is a new minimum distance for SAX. The new similarity measure is fast, it is also tighter and more intuitive than the original one.

5.6.1 Introduction

In Section 5.1.2 we presented SAX, one of the most competitive methods in time series information retrieval. The main advantage of SAX is that the similarity measure it uses, MINDIST, is easy to compute, because it uses statistical lookup tables. In this

section we present an improved similarity measure for SAX. This new measure has the same advantages as MINDIST. But our new similarity measure gives better results in different time series data mining tasks. Its overall complexity is the same as that of SAX, i.e. $O(N)$.

5.6.2 The Updated Minimum Distance (UMD)

The main advantage of SAX, which makes it fast to apply, is that the similarity measure it uses is easy to compute because it is based on pre-computed distances obtained from corresponding lookup tables. However, MINDIST has a main drawback; in order to be lower bounding this similarity measure ignores all the distances between any successive symbols of the alphabet and considers them to be zero. For instance, the lookup table of MINDIST for an alphabet size of 3 is the one shown in Table 5.9. As we can see from the table, all values between any successive symbols are equal to zero. The breakpoints in this case (obtained from statistical tables) are: -0.43 and 0.43. The distance between them is 0.86

Table 5.9 The lookup table of MINDIST for alphabet size equals to 3.

	a	**b**	**c**
a	0	0	0.86
b	0	0	0
c	0.86	0	0

This drawback has two consequences; the first is that MINDIST is not tight enough, which produces many false alarms. The second consequence can be shown by the following example: let the symbolic representing of the five time series $S1, S2, S3, S4, S5$ using *SAX* with alphabet size= 4 be:

$$S1 = aabdd,\ S2 = bacdc,\ S3 = abbcd,\ S4 = bacdd,\ S5 = bbbdc.$$

The MINDIST between any two of these five time series is equal to zero, which is not only unintuitive, since no two time series of these five are identical, but this also produces many false alarms. □

In this section we present a modified minimum distance, which remedies the above problems. The new minimum distance has all the advantages of the original distance, in that it is also a lower bounding of the Euclidean distance, and it is also fast to compute as it uses pre-computed distances. But the new minimum distance is tighter. It is also intuitive because it does not ignore the distances between successive symbols.

The principle of our new minimum distance, which we call the *Updated Minimum Distance* (UMD) is to recover the distances between any successive symbols, which were ignored in MINDIST. Figure 5.3 shows an example of the ignored distances in the case of alphabet size equals to 3, and which are recovered in UMD. The breakpoints are -0.43 and 0.43. In this case the only non-zero distance according to MINDIST is *dist (a,c)* which is equal to 0.86 (the distance indicated by the dashed arrow). The distances represented by the solid arrows are the distances between the minima or the maxima of all the symbols of the alphabet and the corresponding breakpoint. These distances are ignored in MINDIST, but as we can see they are not equal to zero. So *dist(a,b)*, which was zero in *MINDIST* can be updated to become *value2+value4* , and *dist (b,c)* which was also zero in MINDIST can be updated to become *value1+value3*,and even *dist (a,c)* is updated to become *value4+value 3+value0* (the original value).

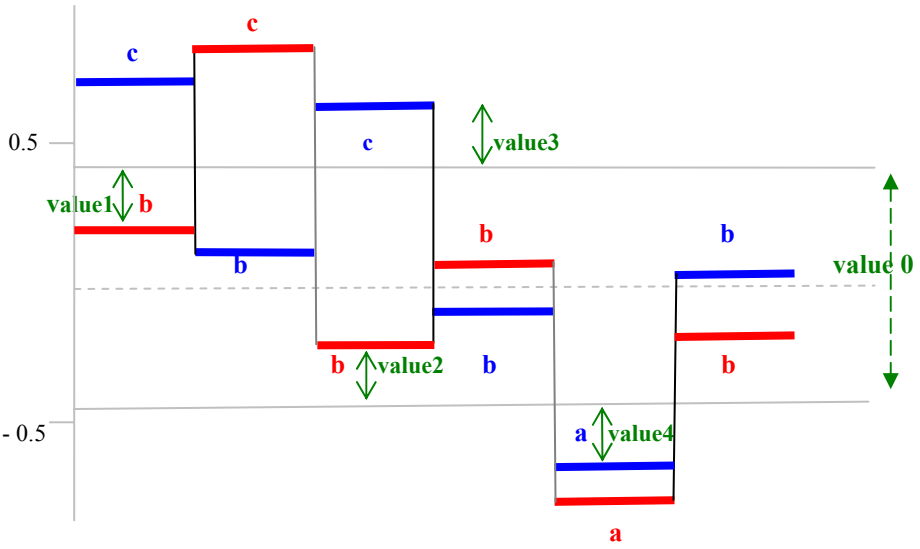


Fig. 5.3. The PAA representation of two time series: *S1 =cbcbab* (boldface, blue) and *S2=bcbbab* (boldface, red). The solid arrows show the ignored distances and the dashed arrow shows the only distance considered by MINDIST : *dist (a,c)=value0* (0.86)

The lookup tables of different alphabet sizes are updated in a like manner. Obviously, this update of lookup tables results in a tighter similarity distance. For instance, the lookup table shown at the beginning of this section can be updated to become the one shown in Table 5.10.

Table 5.10 The updated lookup table for alphabet size equals to 3. We can see that the distances between successive symbols are no longer equal to zero, and the distance *dist (a,c)* is tighter than *dist (a,c)* in Table 5.9

	a	**b**	**c**
a	0	value2+value4	0.86+ value4+value3
b	value2+value4	0	value1+value3
c	0.86+ value4+value3	value1+value3	0

We can easily notice that this new similarity measure is lower bounding of the PAA similarity measure presented in Section 3.4.4, since we take the closest distance between two successive symbols among all the distances of all the PAA segments of these two symbols. As a result, our new similarity measure is also a lower bounding of the Euclidean distance (see Section 3.4.4). This is the same property that MINDIST has.

The other consequence of this update is that UMD, which is based on the updated lookup tables, is intuitive because it gives non-zero values to successive symbols, so UMD of any two of the five time series presented at the beginning of this section is not zero, which is what we expect from any similarity measure applied to these time series because they are not identical.

In order to obtain the minimum and the maximum values of each symbol, the SAX algorithm is modified so that at the step where the different segments of PAA are compared against the breakpoints to decide what symbol is used to discretize that segment, at that step we modify SAX so that it keeps a record of the minimum and maximum values of each segment of that time series. This is performed at indexing time, so it does not include any extra cost at query time. Then when comparing two time series, we take the minimum (maximum) that corresponds to the same symbol of the two times series to find the mutual minimum (maximum) that corresponds to each symbol. These minima and maxima, which are computed at indexing time, are used to update the lookup tables. The updating process includes very few addition operations (three for alphabet size= 3, for instance), whose computational cost is very low compared with the cost of computing the similarity measure. So UMD has the same complexity as that of MINDIST. The pseudo code for the UMD is shown in Figure 5.4.

So, as we can see, the computational cost of UMD is a little bit higher than that of MINDIST at indexing time, but it has the same complexity as MINDIST at query time

We also have to mention that UMD is also a similarity measure and not a distance metric. However, it is one-step closer to being a distance metric since it violates only one condition of the distance metric which is the triangle inequality condition.

```
procedure dist=UMD(S1,S2,S1Min, S1Max,S2Min, S2Max,
   Alphabet, LookupTable)
// INPUT : S1,S2; two input times series presented
// symbolically
// INPUT : S1Min,S2Min; the distance between
// between the minimum value of S1 (S2) and the
// corresponding breakpoint
// INPUT : S1Max,S2Max; the distance between
// between the maximum value of S1 (S2) and the
// corresponding breakpoint
// INPUT : Alphabet
// INPUT : LookupTable is the look up table used with
// MINDIST
// OUTPUT : RETURN dist, the UMD distance between TS1,
// TS2
   for α_k ∈ Σ
      mn= min (S1Min(α_k), S2Min(α_k))
      mx= min (S1Max(α_k), S2Max(α_k))
//update the lookup table for symbol α_k
      UpdateTable ← Update(Lookuptable, α_k, mn, mx)

   end
  return dist=MINDIST(S1, S2, UpdateTable)
end procedure
```

Fig. 5.4. Pseudo code for UMD

5.6.3 Empirical Evaluation

We conducted extensive experiments on the proposed similarity measure. In our experiments we tested UMD on all the 20 datasets available at [190] and for all alphabet sizes which vary between 3 (the least possible size that was used to test MINDIST) to 20 (the largest possible alphabet size). The size of these datasets varies between 28 (*Coffee*) and 6164 (*Wafer*). The length of the time series varies between 60 (*Synthetic Control*) and 637 (*Lightning-2*). So these data sets are very diverse. We also tested these datasets using the Euclidean distance as a reference.

Tightness

As mentioned in Section 2.7, tightness of similarity distances enhances the search process, because it minimizes the number of false alarms. As a result, it decreases the post processing time.

We compared the tightness of UMD with that of MINDIST, for all the datasets and for all values of the alphabet size. In all the experiments, UMD was tighter than MINDIST. In Figure 5.5 we present some of the results we obtained for alphabet size equals to 3 and 10, respectively. We chose to report these datasets because they are the largest data sets in [190], thus their tightness, which is the average of the rate of the corresponding similarity measure to the Euclidean distance between all pairs of time series in the dataset, is more significant statistically.

The experiments conducted on these datasets using other values of the alphabet size, in addition to the experiments on the other datasets in [190] for all values of the alphabet size, all gave similar results.

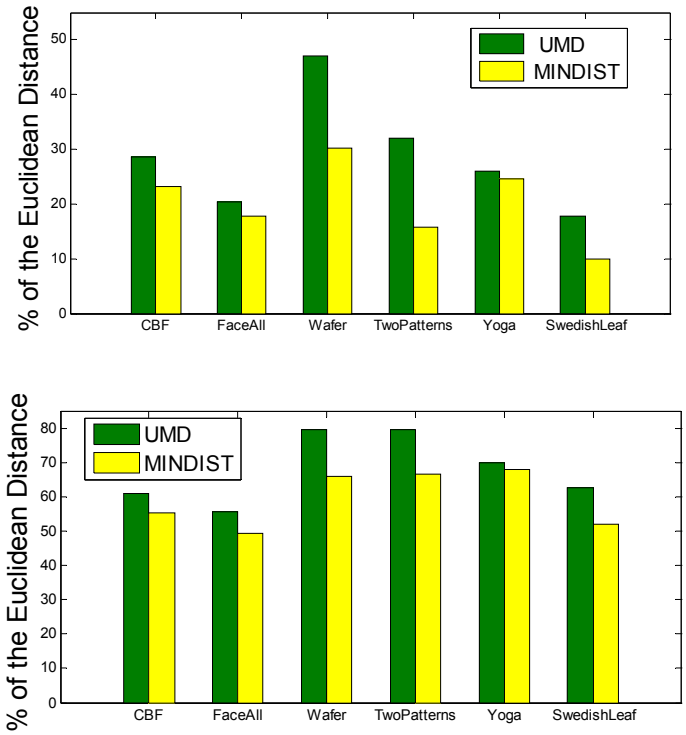


Fig. 5.5. Comparison of the tightness of UMD with the tightness of MINDIST on 6 data sets and for alphabet size=3 (above) and alphabet size=10 (below). The figure shows that UMD is tighter than MINDIST.

Classification

Classification is one of the main tasks in time series data mining. We tested the proposed similarity measure on a classification task on all the data sets available at [190]. We used leaving-one-out cross validation.

In order to make a fair comparison, we used the same compression ratio (the number of points used to represent one segment in PAA) that was used to test SAX with MINDIST (i.e.1 to 4). The first version of SAX used alphabet size that varies between 3 and 10. Then in a later version the alphabet size varied between 3 and 20. We conducted two main classification experiments; the first one is for alphabet size (3:10), and the second is for alphabet size (3:20). Each experiment tested all the datasets. The protocol is the same that we followed in our other experiments on the extensions of the edit distance: in each experiment we start by varying the alphabet size (3 through 10 in the first experiment and 3 through 20 in the second one) on the training set of each dataset to find the optimal value of the alphabet size of that dataset; i.e. the value that minimizes the classification error rate. Then we use that optimal alphabet size on the testing set of that dataset. Table 5.11 shows the results of our experiments for alphabet size in the interval (3:10). We reported the results of the Euclidean distance for comparison reasons (There is no training phase for the Euclidian distance). The best result between UMD and MINDIST is boldfaced and shaded. As mentioned before, the Euclidean distance can sometimes give better results because it is applied to the original uncompressed data.

Table 5.11. The error rate of UMD and MINDIST for α between 3 and 10.

	1-NN Euclidean Distance	**UMD** (α between 3 and 10)	**MINDIST** (α between 3 and 10)
Synthetic Control	0.12	**0.007**	0.033
Gun-Point	**0.087**	**0.213**	**0.233**
CBF	**0.148**	**0.131**	**0.104**
Face (all)	**0.286**	**0.306**	**0.319**
OSU Leaf	**0.483**	**0.471**	**0.475**
Swedish Leaf	**0.213**	**0.291**	**0.490**
50words	**0.369**	**0.338**	**0.327**
Trace	**0.24**	**0.34**	**0.42**
Two Patterns	**0.09**	**0.076**	**0.081**
Wafer	**0.005**	**0.004**	**0.004**
Face (four)	**0.216**	**0.273**	**0.239**
Lighting-2	**0.246**	**0.230**	**0.213**
Lighting-7	**0.425**	**0.411**	**0.493**
ECG200	**0.12**	**0.11**	**0.09**
Adiac	**0.389**	**0.634**	**0.903**
Yoga	**0.170**	**0.193**	**0.199**
Fish	**0.217**	**0.366**	**0.514**
Beef	**0.467**	**0.367**	**0.533**
Coffee	**0.25**	**0.179**	**0.464**
Olive Oil	**0.133**	**0.367**	**0.833**

The results show that for alphabet size in the interval (3:10), UMD outperforms MINDIST.

In Table 5.12 we show the results of our experiments for alphabet size in the interval (3:20).

Table 5.12. The error rate of UMD and MINDIST for α between 3 and 20

	UMD (α between 3 and 20)	**MINDIST** (α between 3 and 20)
Synthetic Control	**0.003**	0.023
Gun-Point	**0.14**	**0.127**
CBF	**0.054**	**0.073**
Face (all)	**0.305**	**0.305**
OSU Leaf	**0.471**	**0.475**
Swedish Leaf	**0.242**	**0.253**
50words	**0.345**	**0.334**
Trace	**0.27**	**0.35**
Two Patterns	**0.065**	**0.066**
Wafer	**0.004**	**0.004**
Face (four)	**0.273**	**0.239**
Lighting-2	**0.229**	**0.148**
Lighting-7	**0.411**	**0.425**
ECG200	**0.11**	**0.13**
Adiac	**0.494**	**0.867**
Yoga	**0.172**	**0.181**
Fish	**0.257**	**0.263**
Beef	**0.333**	**0.433**
Coffee	**0.071**	**0.143**
Olive Oil	**0.3**	**0.833**

The results show that in the interval (3:20) UMD outperforms MINDIST too.

The results obtained for both ranges of alphabet size show that the general performance of UMD is better than that of MINDIST

5.6.4 Discussion

In this section we presented a new similarity measure for SAX. The new similarity measure UMD improves the performance of SAX compared with the original similarity measure MINDIST used with SAX. We conducted several experiments of times series data mining tasks. The results obtained show that SAX with UMD gives better results than SAX with MINDIST. Another interesting feature of the new similarity measure is that it has the same complexity as that of MINDIST.

5.7 Conclusion

In this chapter we presented two main directions of contribution of this dissertation. The first one concerns extending the edit distance. We showed that the main drawback of the ordinary edit distance is that it is based on local procedures, which makes it unable to detect global similarities. We presented another distance metric EED which remedies this limitation by adding a global feature to the edit distance. We tested this new distance in the context of time series data mining on a 1-NN classification task. We showed how EED outperformed the edit distance. We also showed another application of our proposed distance by another author where our proposed distance is applied in the context of natural language processing.

We also presented MREED, which is another extension of the edit distance that considers the frequencies of substrings in addition to those of single characters. Again we showed experimentally that this metric distance gives better results in 1-NN time series classification task than the edit distance. Later we presented Σ_{GRAM} which is a generalization of the previous distances.

In a later section we presented another extension of the edit distance which is PFEED. The main characteristic of this extension is that it is parameter-free. This feature makes PFEED mainly applicable when there are no training datasets. We showed how this extension gives better results in the experiments we conducted than the edit distance.

In the last section of this chapter we presented the second main contribution of this chapter which is a new minimum distance for SAX called UMD to replace MINDIST; the original minimum distance of SAX. We showed through extensive experiments that UMD is not only fast, which is the main advantage of MINDIST, but it is also tighter and more intuitive than MINDIST.

5.8 What Next?

The edit distance has been for long the main distance to compare two strings. This distance is based on a character-to-character comparison. We think this distance is not suitable for comparing new symbolic data types. In comparing two DNA or RNA sequences, the strings are usually very long, and the alphabet size is small. This requires certain modifications on the edit distance, like considering the frequencies in a way similar to or different from the approach we used in our proposed distances, instead of the ordinary atomic edit operations.

The reversal operation is a fundamental operation in some applications like gene rearrangement [166], [115], yet the ordinary edit distance is unable to handle this operation except through long, indirect edit operations.

In Natural Language Processing (NLP) we can think of a generalized edit distance that compares sentences instead of words. In this case the alphabet is much larger than that used in the edit distance. This generalized distance can also include a statistical feature that considers the letter or groups of letters that usually follow a certain letter in a certain language.

In some applications like auto completing we see that many algorithms can handle spelling errors sufficiently when the user mistypes the last part of the entry , and can still correctly predict the word the user is trying to type, while they seem to be completely inefficient when the user mistypes the first part of the entry. We think this can be handled using an edit distance that assigns costs to the edit operations not only based on the operation itself, but also on the location of the characters in the two strings that a delete operation, for instance should be assigned a higher cost if the deletion takes place at the beginning of the two strings than at the end of them.

Although the symbolic representation method presented in this chapter is a substantial improvement over the first symbolic methods, we think there is a lot of research to be done in this field of computer science. In fact, although the main motive behind these methods was to benefit from bioinformatics and textual data mining algorithms, we did not see any approach to benefit from these algorithms. In many cases, the symbolic representation was expressed numerically again to perform a required operation. We understand symbolic representation as an equivalent to the numeric representation and that it permits defining symbolic operations without having to resort to the corresponding numeric representation of the data. The main advantage of this equivalent representation is that some tasks are better handled numerically (similarity search, classification, median strings, etc) while other tasks are symbolic in nature (motif discovery, anomaly detection, etc). We think an ideal representation of data should be numerical-symbolic in a way that all operations in one representation have their equivalence in the other. Such a representation can tackle different tasks in data mining more efficiently.

However, the main challenge to introducing such a representation is that symbolic representation usually implies loss of information which is not easily recovered with most of the symbolic representation techniques that we have now. We think that some mathematical tools can be helpful to overcome this obstacle.

Chapter 6

Multi-resolution Approaches to Time Series Indexing and Retrieval

In this chapter we present another major contribution of this dissertation which is our work on multi-resolution methods in time series indexing and retrieval which we presented in [130], [132], [131], and [136].

This chapter is organized as follows: Section 6.1 is a background section in which we present the principle of multi-resolution as tackled in several domains of multimedia. We mainly focus on how it has been exploited in the time series data mining and information retrieval communities. The first contribution of this chapter: MIR, which is a new standalone multi-resolution algorithm, is presented in Section 6.2. The filtering mechanism on which MIR is based is presented in Section 6.2.3 and the algorithm itself is described in Section 6.2.4. Section 6.2.5 is the experimental section in which we show the results of the experiments we conducted on MIR. The second contribution of this chapter is a novel multi-resolution algorithm that we combine with a dimensionality reduction technique to enhance its performance. This algorithm called MIR_X is introduced in Section 6.3, described in Section 6.3.3, and validated experimentally in Section 6.3.4. The third contribution of this chapter is an improvement of the two previously stated algorithms. This improved algorithm, called Tight_MIR, has the advantages of both MIR and MIR_X. Tight_MIR is presented in Section 6.4. Extensive experiments on this improved algorithm are presented in Section 6.4.3. We conclude this chapter with future perspectives in Section 6.5.

6.1 Multi-resolution Methods in Multimedia Data Mining

Multi-representation approaches physically store data at different scales in a database [68]. These levels are called *resolution levels*. In these approaches data are pre-generated and stored at different resolution levels. The principle of this representation is that a representation with a maximum resolution contains all data of the lower resolutions [184]. *Multi-resolution* approaches store only the data at the highest level of resolution and simplify and generalize data dynamically [217].

Multi-resolution methods are widely used in multimedia databases. In geographical databases, multiple representations and multiple resolutions are used in the framework of a project [150] to enhance *Geographic Information System* (GIS). The project supports multiple resolutions of geographic data.

Image retrieval is another field of multimedia in which multi-resolution methods are used. In [187] the authors use a wavelet transform to obtain a multi-resolution representation of the searched shape based features. In [80] the authors propose a multi-resolution multi-grid framework for image retrieval. This framework uses color, texture and shape features. The images are partitioned into non-overlapping tiles, while the texture and color features are extracted from these tiles at two different resolutions in two grid framework. Multi-resolution *Matching Pursuit* is used in [67] to decompose images. The authors propose a multi-resolution strategy to reduce encoding complexity. Multi-resolution is also used for a color reduction algorithm in [159]. The algorithm is based on color distribution and on the use of multi-resolution image representation.

In [198] the authors use multi-resolution schemes to estimate missing values for DNA micro-arrays. The basis of this method in derived from the principle of multi-resolution analysis; undetected characteristic at one resolution may easily be spotted at another. In their work the authors investigate the scheme of second order wavelets known as *lifting schemes* as a method for estimating missing data in cDNA (complementary DNA) micro-array experiments.

Multi-resolution methods have also been exploited in time series information retrieval and data mining. In [21] a visualization application for very large multidimensional time series datasets is developed. The proposed data model supports multiple integrated spatial and temporal resolutions of the original data. The system incorporates an indication of the error introduced by the multi-resolution data representation into the visualization. So these regions of low resolution where the error is high can then be explored again using higher resolutions. Using multi-resolution techniques to effectively visualize large time series is also applied in [76] where the proposed framework uses multiple resolution levels. The basic idea of this framework is to allocate space in proportion to the degree of interest of data subintervals. This strategy enables the user to perceive important information, and it also frees required display space to visualize all the data.

In time series data mining multi-resolution approaches have been used in motif discovery problems. In [34] the authors propose a method based on the multi-resolution property of *i*SAX [174] and [173] to derive motifs at different resolutions. This enables the user to navigate in the Top-K motifs hierarchy structure to better understand the time series database at hand.

Clustering, an important problem in time series data mining, is another domain where multi-resolution methods have been used. The authors of [114] propose a multi-resolution PAA to achieve an algorithm for iterative clustering. This clustering process is sped up by examining the time series at increasingly higher resolution levels of the PAA. Stopping criteria are proposed to decide how many levels are needed. The authors use this algorithm for streaming time series. In [197] and [113] the authors propose a time series k-means clustering algorithm based on the multi-resolution property of wavelets. In the proposed algorithm an initial clustering is performed using a very coarse representation of the data. The results of this clustering

are then used to initialize another clustering at a higher resolution level. This process is repeated several times until the results of the clustering stabilize. The advantage of this algorithm is that it permits the user to terminate it at any level. The authors apply their algorithm to images utilizing two descriptors: color and text, and treating them as time series.

In [126] and [200] a method of multi resolution representation of time series is presented. This symbolic method uses a multi-resolution vector quantized approximation of the time series together with a multi-resolution similarity distance. Using this representation the method keeps both local and global information of the time series data.

Our approach, however, is substantially different from the other methods in that it aims to speed up the similarity search by reducing query-time distance evaluations to the least degree possible. This is achieved in our approach by using two techniques; the first is applying fast-and-dirty filters based on pre-computed distances, the second is that when distance evaluations are inevitable, our approach computes the required distance at lower resolutions where the cost of the distance evaluation is low, and even when moving to a higher resolution and having to re-compute the distance our approach recycles computations from lower resolutions to compute the distances at higher resolutions.

6.2 The Multi-resolution Indexing and Retrieval Algorithm -Weak MIR

In this section 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.

6.2.1 Introduction

Given a query Q, a radius r, and a time series database U. Time series representation methods that we presented in Chapter 3 process this similarity query using the following algorithm:

1- Choose a lower dimensional space.

2- Represent the time series in the reduced space.

3- Define a lower bounding similarity distance on this reduced space.

4- Process the similarity search in the reduced space.

5- Exclude the time series which are farther than r from the query and return a *candidate answer set*.

6- Scan this candidate answer set using the original time series and the original similarity distance and return 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 necessarily 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, corresponding 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 computational complexity at any level is always less than that 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* method (MIR). Notice that this version of our method does not require additional conditions to apply to indexing time series (unlike the version we present in Section 6.3) so in mathematical terms this version is "weak".

6.2.2 Concepts and Terminology

Let O be the original n-dimensional space where the time series are embedded, R is a $2m$-dimensional space, where $2m \leq n$. 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 at indexing-time.

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 6.1 illustrates the different definitions we presented in this section. The segment $[t_0, t_3]$ 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 $[a, d]$.

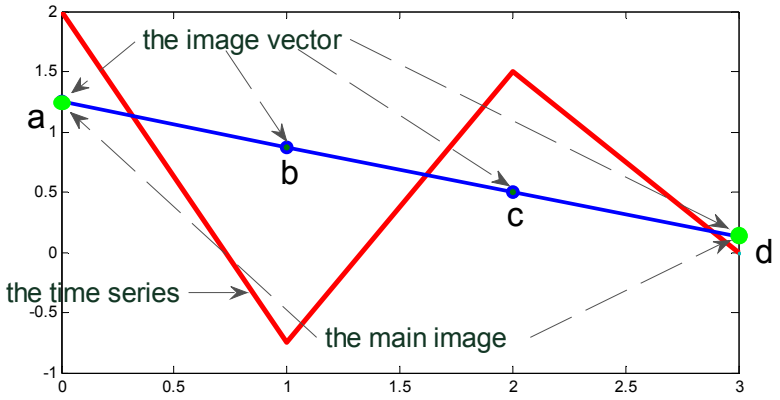


Fig. 6.1. The different concepts of the proposed method

Figure 6.2 shows how successive segments are represented. We notice in this figure that b_1 the right main image of the first segment is different from a_2 the left main image of this second segment. This is a main property of our representation.

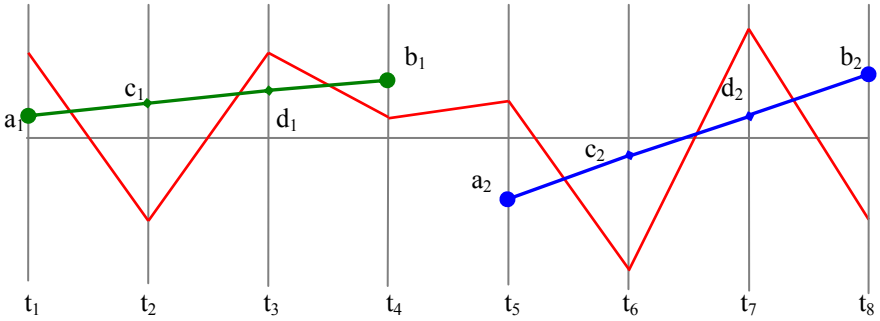


Fig. 6.2. The image vectors and the main images of two successive segments.

We define two distances on the representation space; 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. $d(S_i, \widetilde{S}_i)$. 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:

$$d^R(S_i^R, S_j^R) \le d^R(\widetilde{S}_i, \widetilde{S}_j) \tag{6-1}$$

Relation (6-1) 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 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 $\widetilde{S}^{(k)}$ and $S^{R(k)}$, respectively. Figure 6.3 shows an illustration of the relationships between the previous concepts.

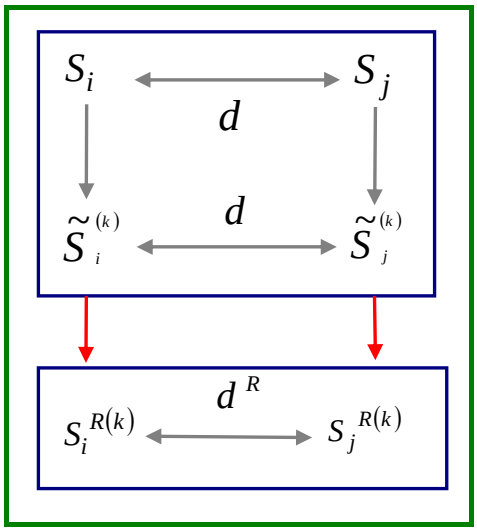


Fig. 6.3. The original space (O,d) embeds the original time series S_i, S_j (top), the images of the approximated time series $\tilde{S}_i, \tilde{S}_j$ (middle). The reduced space R embeds the main images of the approximated time series S_i^R, S_j^R (bottom).

6.2.3 The Double Filtering Inequalities

Given a range 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 triangle inequality we get:

$$d\left(\tilde{Q}^{(k)},S\right) \le d(Q,S) + d\left(Q,\tilde{Q}^{(k)}\right) \quad \forall S \in O \tag{6-2}$$

So now the range query can be expressed as:

$$d\left(\tilde{Q}^{(k)},S\right) \le r + d\left(Q,\tilde{Q}^{(k)}\right) \tag{6-3}$$

Since $\tilde{S}^{(k)}$ is the best approximation of S at level k, then for any $S \in O$ we have:

$$d\left(\tilde{Q}^{(k)},S\right) \ge d\left(S,\tilde{S}^{(k)}\right) \tag{6-4}$$

So (6-3) can be expressed as:

$$d\left(S,\tilde{S}^{(k)}\right) \le r + d\left(Q,\tilde{Q}^{(k)}\right) \tag{6-5}$$

This means that all the data objects that satisfy:

$$d\left(S,\tilde{S}^{(k)}\right) > r + d\left(Q,\tilde{Q}^{(k)}\right) \tag{6-6}$$

Should be excluded.

In a similar way, by applying the triangle inequality again, we get:

$$d\left(Q,\tilde{S}^{(k)}\right) \le d(Q,S) + d\left(S,\tilde{S}^{(k)}\right) \tag{6-7}$$

And the range query can be expressed as:

$$d\left(Q,\tilde{S}^{(k)}\right) \le r + d\left(S,\tilde{S}^{(k)}\right) \tag{6-8}$$

Since $\tilde{Q}^{(k)}$ is the best approximation of Q at level k, then for any $S \in O$ we have:

$$d\left(Q,\tilde{S}^{(k)}\right) \ge d\left(Q,\tilde{Q}^{(k)}\right) \tag{6-9}$$

So (6-8) can be expressed as:

$$d\left(Q,\tilde{Q}^{(k)}\right) \le r + d\left(S,\tilde{S}^{(k)}\right) \tag{6-10}$$

This means that all the data objects that satisfy:

$$d\left(Q,\tilde{Q}^{(k)}\right) > r + d\left(S,\tilde{S}^{(k)}\right) \tag{6-11}$$

Should also be excluded.

From both (6-6) and (6-11), we can write:

$$\left| d\left(Q,\tilde{Q}^{(k)}\right) - d\left(S,\tilde{S}^{(k)}\right) \right| > r \tag{6-12}$$

Inequality (6-12) defines the first exclusion condition, which we call *the first filter*.

On the other hand, by applying the triangle inequality again, we get:

$$d\left(\tilde{S}^{(k)},\tilde{Q}^{(k)}\right) \le d\left(\tilde{Q}^{(k)},S\right) + d\left(S,\tilde{S}^{(k)}\right) \tag{6-13}$$

Using the triangle inequality again, and substituting in the above relation we get:

$$d\left(\tilde{S}^{(k)},\tilde{Q}^{(k)}\right)\leq d(Q,S)+d\left(Q,\tilde{Q}^{(k)}\right)+d\left(S,\tilde{S}^{(k)}\right) \tag{6-14}$$

Or:

$$d\left(\tilde{S}^{(k)},\tilde{Q}^{(k)}\right)\leq r+d\left(Q,\tilde{Q}^{(k)}\right)+d\left(S,\tilde{S}^{(k)}\right) \tag{6-15}$$

If d and d^R are Euclidean, and taking (6-1) into account, we can write:

$$d^R\left(S^{R(k)},Q^{R(k)}\right)\leq d\left(\tilde{S}^{(k)},\tilde{Q}^{(k)}\right) \tag{6-16}$$

By substituting in (6-15) we get the second exclusion condition:

$$d^R\left(S^{R(k)},Q^{R(k)}\right)> r+d\left(Q,\tilde{Q}^{(k)}\right)+d\left(S,\tilde{S}^{(k)}\right) \tag{6-17}$$

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

6.2.4 The Algorithm Description

At 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 for convenience. The segmenting is recursive so all the points of a certain level are included in a higher level. The shortest length corresponds to the highest level, and the longest corresponds to the lowest level.

Next 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, we use polynomials for their simplicity.

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

At query-time: The query is divided into segments with the same lengths as those of the indexed 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 at indexing-time. The distances $d\left(Q,\tilde{Q}^{(k)}\right)$ are computed. Notice that $d\left(Q,\tilde{Q}^{(k)}\right)$ 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 pre-computed at indexing-time. The second filter contains two

distances that have been computed at indexing-time ($d\left(Q,\tilde{Q}^{(k)}\right),d\left(S,\tilde{S}^{(k)}\right)$), so the only distance that is to be computed at query-time is $d^R\left(S^{R(k)},Q^{R(k)}\right)$. 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 (6-12). If this time series is excluded, we move to the next time series, if not, we try to exclude this time series using relation (6-17). If all the time series in the database have been excluded the algorithm terminates immediately, 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 failed to exclude the time series using a less expensive distance at a lower resolution level.

6.2.5 Experiments

We conducted extensive experiments on different datasets available at [190] in a similarity search problem. Because scalability is a desired property in similarity search algorithms, and 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 time series. The length of the time series varied 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.

Although several papers present experiments based on wall clock time, it is a poor choice and subject to bias [92], [57] so we preferred to use another method in the all the experiments we conducted in this chapter.

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 [167]. 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 took 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 6.4, 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). The results show that MIR outperforms sequential scanning by an order of magnitude on average.

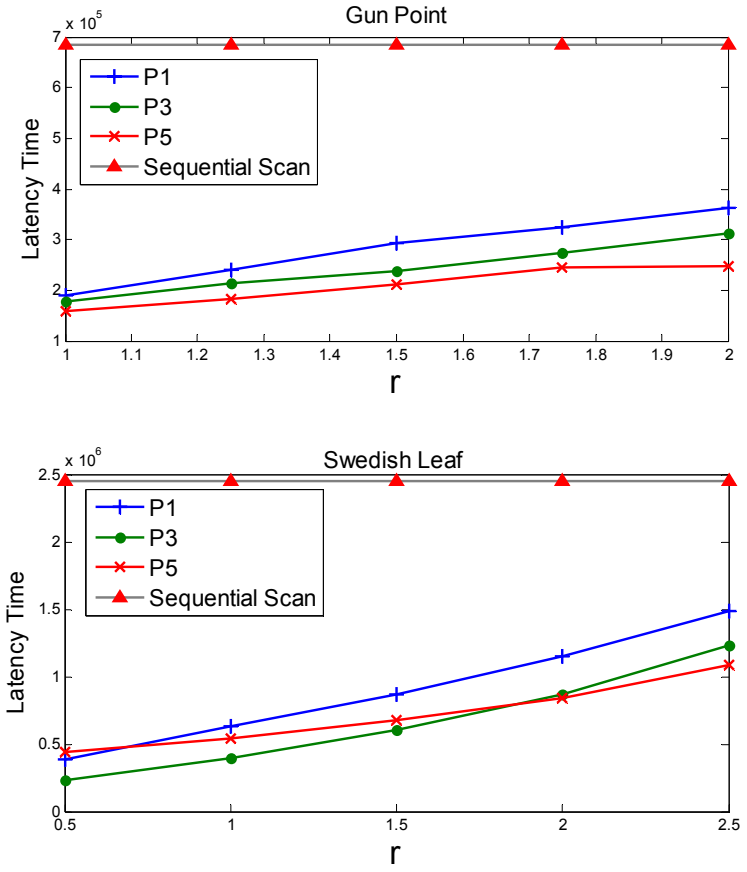


Fig. 6.4. Comparison of the latency time between sequential scanning and MIR on datasets (*Gun Point,* length=150) (above), and (*Swedish Leaf,* length=128) (below). The figure shows the latency time using as an approximating function a polynomial of the first (P1), third (P3), and fifth degree (P5)

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 6.5 shows the results we obtained with the two longest time series among the tested datasets (*Yoga*, length=426) and (*Fish*, length=463).

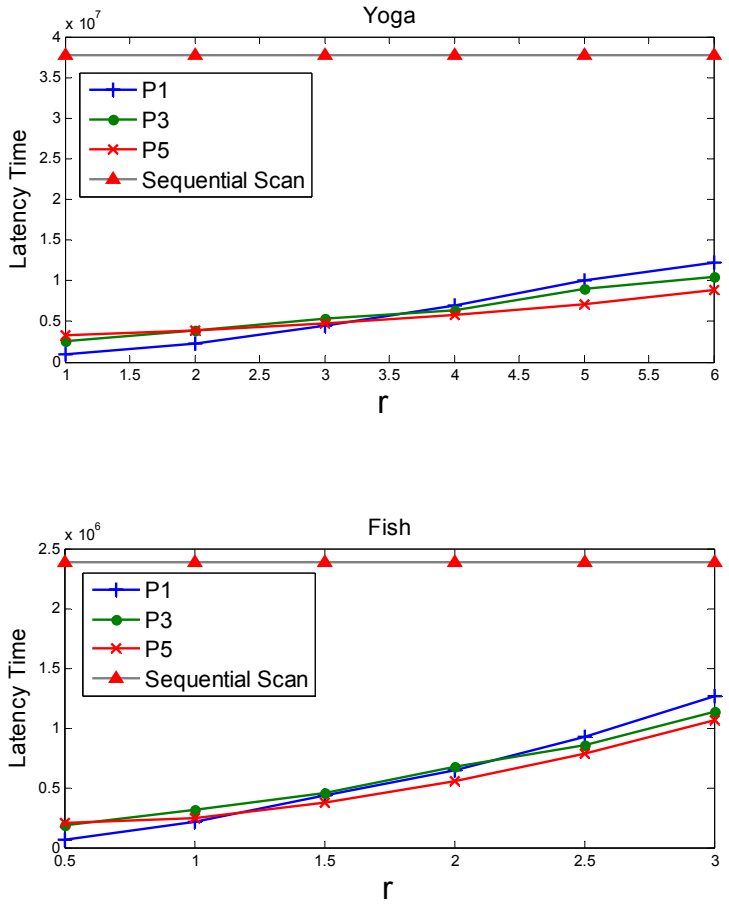


Fig. 6.5. The performance of the two longest datasets (*Yoga*) and (*Fish*)

We can also see from Figure 6.5 that as the time 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,* length=1500, from [156]). This dataset is almost four times as long as the datasets at [190]. Figure 6.6 shows the results we obtained from applying MIR to this dataset.

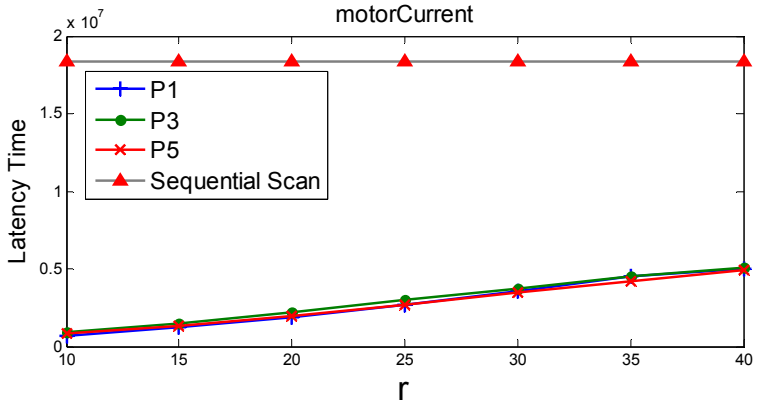


Fig. 6.6. Comparison of the latency time between sequential scanning and MIR on dataset (*motorCurrent*).

These results enhance the two previously stated outcomes that the performance gets better in general as the time series get longer, and 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 called (*Tickwise*) (from [182]). This dataset consists of one very 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 6.7 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 6.7 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 on 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 6.7 are superposed.

The results clearly show that the performance of MIR improves as the length of the time series gets longer for this dataset.

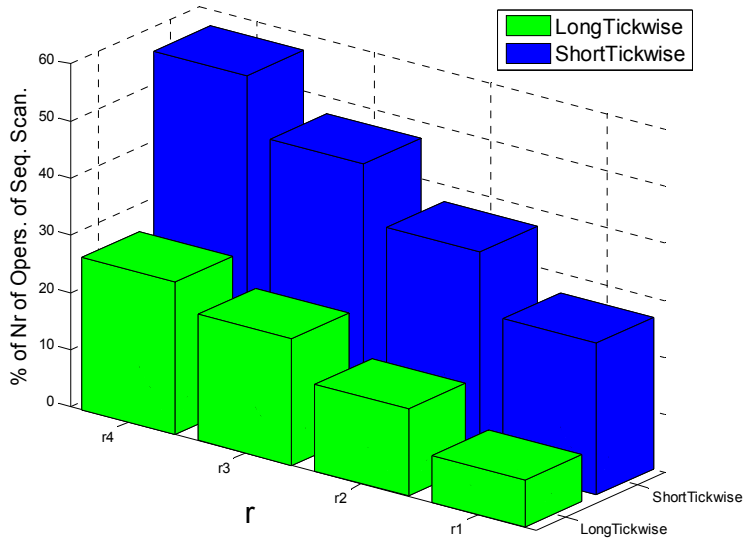


Fig. 6.7. Comparison of the proportion of operations of (*LongTichwise*) to the number of operations of sequential scanning of (*LongTichwise*) with the proportion of operations of (*ShortTickwise*) to the number of operations of sequential scanning of (*ShortTichwise*). The approximating function is 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 6.8 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.

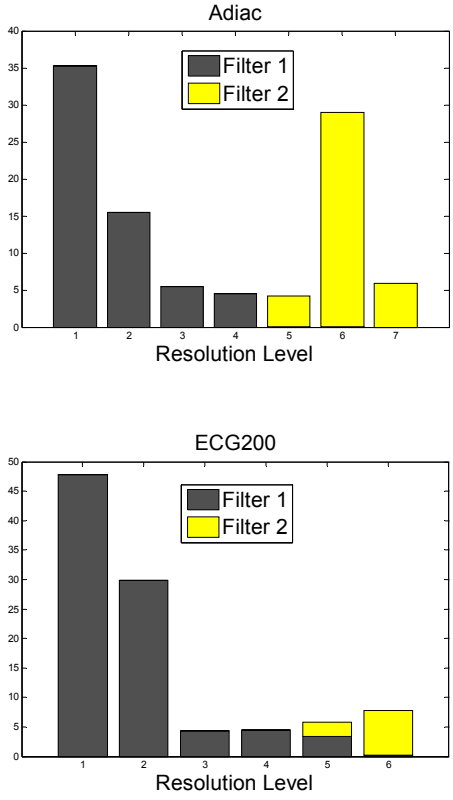


Fig. 6.8. The exclusion process of datasets (*Adiac*) (above) and (*ECG200*) (below) for r that returns 1% of the time series.

It is important to mention that Figure 6.8 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 the first filter and it applies the second filter only on the time series that could not be excluded by the first filter. This in fact means that the second filter has a higher exclusion power than the first filter. 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 6.8, the exclusion effect for small values of r results mainly from the first filter. We can also see that, in general, the exclusion power of the first filter decreases as the resolution level gets higher. The exclusion power of the first filter also decreases when r gets larger as we can see from Figure 6.9 which

shows the exclusion process of the same datasets presented in Figure 6.8 but for values of r that return 10% of the time series

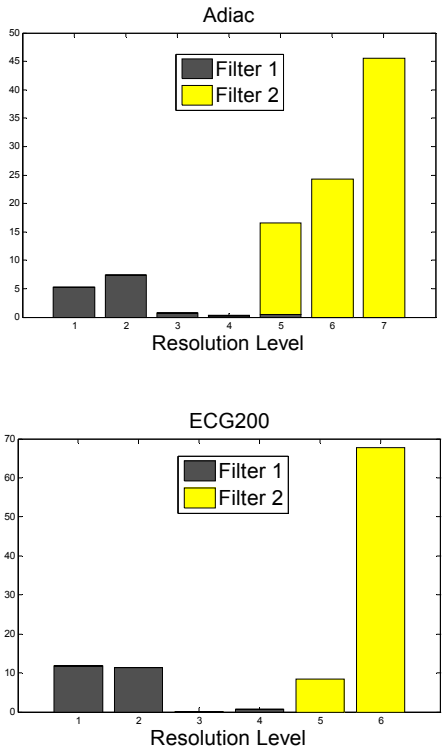


Fig. 6.9. The exclusion process of datasets (*Adiac*) (above) and (*ECG200*) (below) for r that returns 10% of the time series.

Several different heuristics can improve the performance of our method. For instance, sorting the distances before applying the exclusion conditions reduced the number of operations required to perform the similarity query. Figure 6.10 shows a comparison of this heuristic applied to dataset (*Two Patterns*) with an approximating polynomial of the first degree.

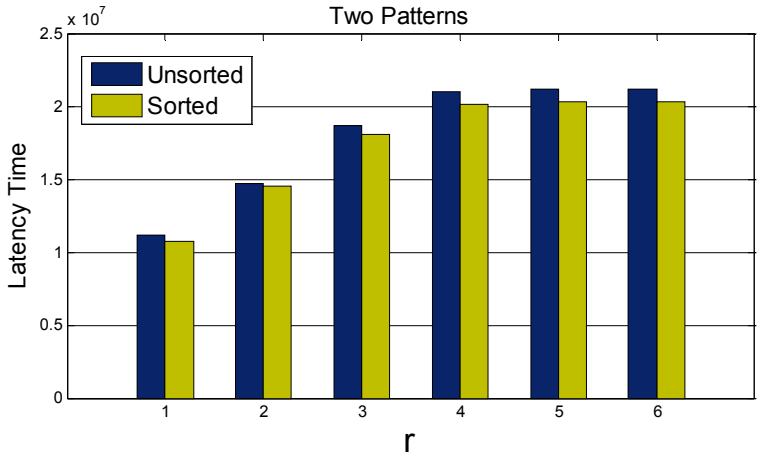


Fig. 6.10. Comparison of the latency time of MIR applied with and without sorted distances on dataset (*Two Patterns*) using a first degree polynomial as an approximating function

6.2.6 Remarks on the Filtering Process

The experiments show that the performance of the two filters seems to be complementary, since the first filter filters out more time series at lower resolution levels, while the second filter filters out more time series at higher levels. This phenomenon can be explained by examining relations (6-12) and (6-17): at lower resolution levels the segments are longer, so the approximation error is higher. As a consequence, the absolute difference in the first filter is a difference between relatively large numbers: $d\left(Q,\tilde{Q}^{(k)}\right)$, $d\left(S,\tilde{S}^{(k)}\right)$,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 the second filter is different: at lower levels $d^R(S^{R(k)},Q^{R(k)})$ is small while $d\left(Q,\tilde{Q}^{(k)}\right)+d\left(S,\tilde{S}^{(k)}\right)$ 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 larger, while $d\left(Q,\tilde{Q}^{(k)}\right)+d\left(S,\tilde{S}^{(k)}\right)$ 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 MIR is complex as 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 6.2.4, this depends on the dataset, the resolution level and the value of r.

6.2.7 Discussion

We presented a new algorithm to tackle the similarity search problem. The proposed algorithm is based on a fast-and-dirty filtering scheme that iteratively reduces the search space. For each resolution level the time series are represented by an appropriate approximating 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 true answers to the query. A post-processing scanning on this candidate response set is performed to filter out any false alarms and return the final response set. The conducted experiments on 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 section we presented the results obtained by using the Euclidean distance, but our method can support a variety of distances. We conducted other experiments using L_1 distance and they gave similar results.

6.3 Combining a Multi-resolution Filter with a Representation Method-Strong MIR

In this section we introduce another version of our multi-resolution method that does not function autonomously but by accompanying a representation method. We show how an algorithm based on coupling the principle of multi-resolution fast-and-dirty filtering we presented in Section 6.2 with a dimensionality reduction technique can boost its performance. Since this algorithm requires that the dimensionality reduction method be lower bounding to the original similarity distance on the raw data (which is practically the case with most dimensionality reduction techniques) our method is mathematically "strong". We call our method the *Multi-resolution Indexing and Retrieval_X (*MIR_X*)*, where X is the dimensionality reduction technique used.

6.3.1 The Principle

Let O be the original, n-dimensional space where the time series are embedded. Each time series $S \in O$ is divided into N consecutive segments, where $N = 2m$ is the dimension of the reduced space, i.e. the space that the dimensionality reduction technique uses. Each segment $[t_i, t_j]$ of this time series is approximated by a function of low dimension in the same way we did in Section 6.2.2. We proceed in the same way to get the image vector of the time series in the database.

Let R be the lower N-dimensional space that the dimensionality reduction technique uses, where $N<n$. The time series in the database are represented in the reduced space R using the chosen dimensionality reduction technique.

So now each time series has two representations: the first is an n-dimensional one, by using the approximating function, and the second is an N-dimensional one, by using the dimensionality reduction technique. Figure 6.11 illustrates the different concepts we presented in this section, where the dimensionality reduction technique used is PAA, and the approximating function is a first-degree polynomial.

Our method uses two similarity distances: 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 or the distance between the original time series and its image vector (see Section 6.2.2) The second distance is denoted by d^N, and it is the distance defined by the chosen dimensionality reduction technique.

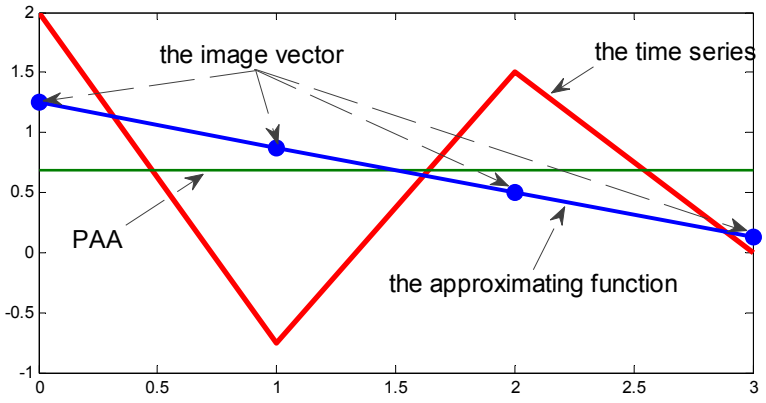


Fig. 6.11. The different concepts of the principle of MIR_PAA (the dimensionality reduction technique used is PAA). The segments are approximated by a first-degree polynomial

6.3.2 The Filtering Process

The two filters described in Section 6.2.3 can be used with MIR_X. However, using the exclusion condition of a dimensionality reduction technique which uses a lower bounding condition to the distance in the original space makes the second filter described in Section 6.2.3 redundant, because it is overwritten by the more powerful exclusion condition of the dimensionality reduction technique.

6.3.3 The Algorithm

At indexing-time: We start by choosing the dimensionality reduction technique to be used, and then we choose the length of segments at each resolution level. Then we choose the approximating function to be used with all the time series and for all resolution levels. We compute and store all the distances $d\left(S,\widetilde{S}^{(k)}\right)\ \forall S \in O$, for all resolution levels the same way we did in Section 6.2.4.

At query-time: Like in Section 6.2.4, the query is segmented at each resolution level using the same lengths of segments that were used to segment the time series. Then these segments are approximated using an approximating function of the same type and degree that was used to approximate the time series. The query is also represented using the same dimensionality reduction technique that was used at indexing-time. The distances $d^N\left(Q^{R(k)},S^{R(k)}\right)$ are computed when needed (only when needed). $d\left(Q,\widetilde{Q}^{(k)}\right)$ is also computed.

At each resolution level, the exclusion condition defined by relation (6-12) is much less costly than the exclusion condition defined by the dimensionality reduction technique, because the first filter does not include any distance computations, since the two distances it uses have already been computed at indexing-time.

Just like with MIR, the algorithm starts at the lowest resolution level by applying the first filter to the first time series. If this first time series is excluded, we move to the next time series and apply the first filter to this second time series, if it is not excluded, the algorithm computes $d^N\left(Q^{R(k)},S^{R(k)}\right)$ for this first time series and at that level and applies the exclusion condition of the dimensionality reduction technique to this first time series, then it moves to the next time series. The algorithm continues in the same manner as that of MIR for higher resolution levels. At the end we get a candidate answer set which is sequentially scanned using the original distance to get the final answer set.

6.3.4 Experimental Validation

We tested our new method using the same the datasets available at [190] in a similarity search problem. We also excluded the datasets that are too small like we did in the experiments in Section 6.2.5. We first tested our method using PAA because this dimensionality reduction technique is widely used and the concept of multi-resolution on this method is straightforward. The objective of our experiments was to see how much our method (MIR_PAA) improves the performance of the dimensionality reduction technique (PAA) when this latter is used as a standalone technique. The compression ratio is 1:4. We chose this compression ratio since it is the compression ratio used with PAA.

We compared the number of operations that (MIR_PAA) needed to perform the similarity search with the number of operations that (PAA), as a standalone method, needed, using the latency time concept. The number of operations that sequential scanning needed was also computed for comparison reasons.

The approximating function we used was a first-degree polynomial. The results were also the average of 100 runs on each dataset.

In the case where N is not a factor of n, the authors of [92] padded the time series with zeros. This approach actually puts our method at a disadvantage, because we have different values of N, which correspond to different resolution levels, so we have to add more zeros. But still, we tested our method using this approach. In Figure 6.12 we present some of the results we got. The results we got using other datasets were similar.

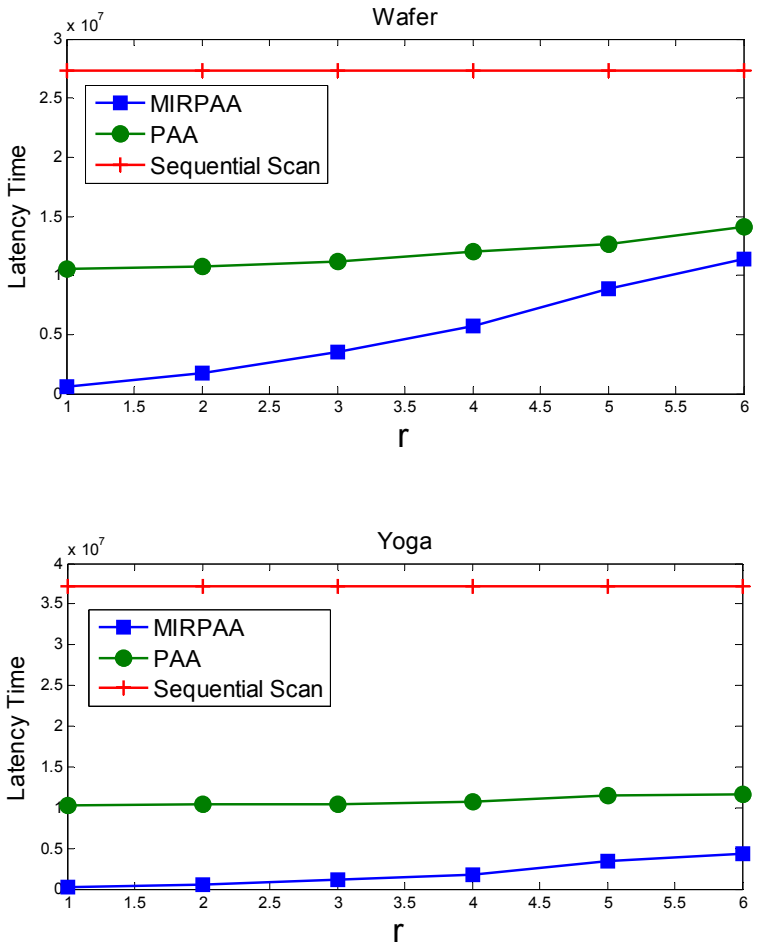


Fig. 6.12. Comparison of the latency times of sequential scanning, MIR_PAA, and PAA, on datasets (*Wafer*) (above), and (*Yoga*) (below). The approximating function is a first-degree polynomial. The time series are padded with zeros

We also conducted other experiments, where the time series were truncated so that N is a factor of n . In this case there is no need to add zeros, so the methods are applied to real data only. We show in Figure 6.13 some of the results we obtained. We obtained similar results when we conducted these experiments on the other data sets.

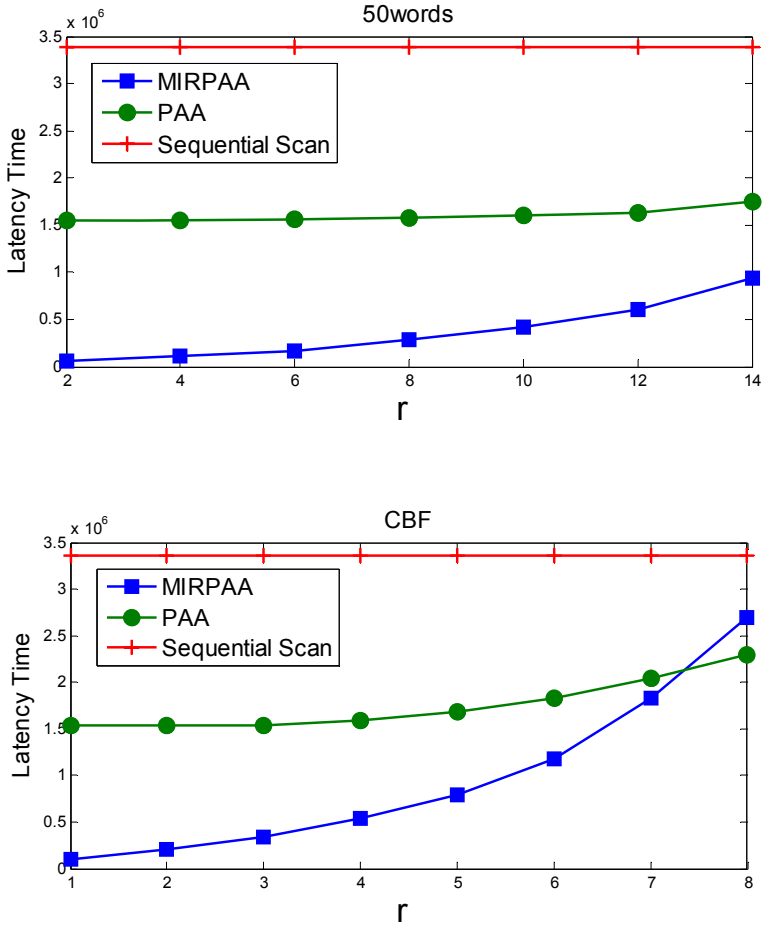


Fig. 6.13. Comparison of the latency times of sequential scanning, MIR_PAA, and PAA, on datasets (*50words)* (above), and (*CBF*) (below). The approximating function is a first-degree polynomial. The time series are truncated

We also tested our method using SAX as a dimensionality reduction technique because it is a fast method, so we wanted to see if our method can still speed up a dimensionality reduction technique which is already fast. As mentioned before, 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 from [190], and for different values of the alphabet size. The codes we used in the experiments were optimized versions of the original codes, since the original codes written by the authors of SAX were not optimized for speed

We report in Tables 6.1 and in Figure 6.14 the results of (*Wafer*). We chose to present the results of this dataset in particular because it is the largest dataset in the repository, also because it is shown in [112] by the authors of SAX that the best results obtained with SAX were with this dataset. 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).

Table 6.1. Comparison of the latency time between SAX and MIR_SAX for r=1:4 and alphabet size=3, 10, 20

	α= 3	α= 10	α= 20
MIR_SAX	1.0592E6	3.7136E5	2.6734E5
SAX	5.1291E6	1.5764E6	1.2759E6

r=1
(**a**)

	α= 3	α= 10	α= 20
MIR_SAX	7.2062E6	3.3509E6	2.2255E6
SAX	1.567E7	4.8078E6	3.4253E6

r=2
(**b**)

	α= 3	α= 10	α= 20
MIR_SAX	1.3717E7	1.1444E7	9.5446E6
SAX	2.1944E7	1.4428E7	1.1144E7

r=3
(**c**)

	α= 3	α= 10	α= 20
MIR_SAX	2.2697E7	1.6928E7	1.6611E7
SAX	2.8287E7	2.2179E7	1.9877E7

r=4
(**d**)

The results obtained show that MIR_SAX outperforms SAX for the different values of r and for the different values of the alphabet size.

The results of testing our method on all the other datasets are similar to the results obtained with (*Wafer*)

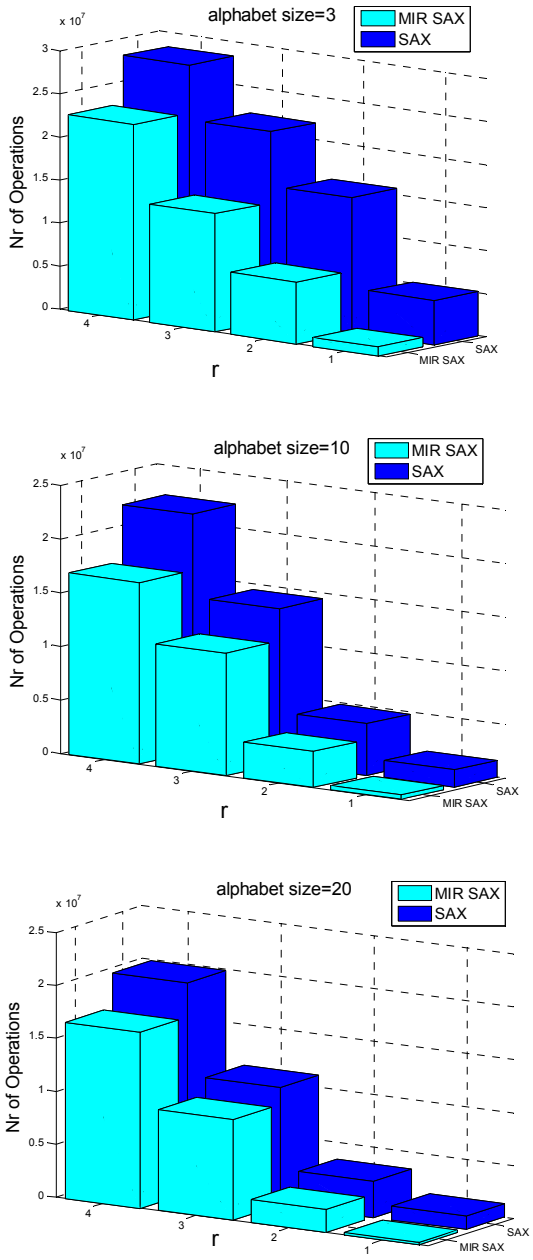


Fig. 6.14. Comparison of the latency time between MIR_SAX and SAX for alphabet size=3, 10, and 20

We wanted to see if training will improve the performance of our method. The basis of this experiment is that we noticed that the performance of MIR_X is related to the value of r, so we wanted to find an optimal scheme to using MIR_X by making the algorithm decide whether to continue with the available resolution levels or to terminate this process and move directly to post-processing when it estimates that the rest of the resolution levels will probably not exclude many time series. The protocol we used for this experiment was as follows: at indexing-time we used more resolution levels than usual (12 levels for (*Wafer*) compared with 6 levels for untrained experiments, of course in this experiment the length of the time series was not a power of 2 but arbitrary). Then for each value of r, we tested all possible combinations of resolution levels that yield the minimum latency time. The dataset we used in the training is (*Wafer_training*) (also from [190]). The optimal combination of resolution levels that corresponds to a certain value of r that we got from training the algorithm on (*Wafer_training*) was used with (*Wafer_test*). Figure 6.15 shows that the latency time of the trained MIR_PAA is shorter than that of untrained MIR_PAA. Interestingly, we see in Figure 6.15 that training is more beneficial as r gets larger.

Training other datasets gave similar results.

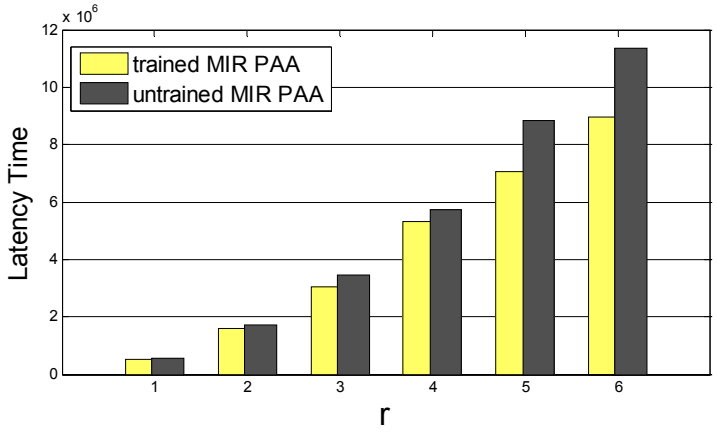


Fig. 6.15. Comparison between the latency time of trained MIR_PAA and untrained MIR_PAA on dataset (*Wafer*)

6.3.5 Discussion

In this Section we presented a new frame to tackle the similarity search problem. The basis of this frame is to combine a dimensionality reduction technique with a multi-resolution fast-and-dirty filter to enhance the pruning power of this technique. We conducted several experiments which show that the proposed frame improves the performance of dimensionality reduction techniques.

Other heuristics can be used to improve our frame, like recycling the computations when we move from one resolution level to another, or sorting the distances before applying the first filter.

In our experiments we tested our method using PAA and SAX because of the reasons we mentioned in Section 6.3.4, but most papers on time series multi-resolution indexing methods use DWT because of the multi-resolution nature of this representation method. We think further investigation on applying MIR with DWT is worth considering.

6.4 An Improved Multi-resolution Indexing and Retrieval Algorithm –Tight MIR

In this section we revisit Sections 6.2 and 6.3 and introduce a third version of our multi-resolution algorithm. This improved version, which we call Tight_MIR has the advantages of both MIR and MIR_X in that it is a standalone method, like MIR, yet it has the same competitive performance of MIR_X.

6.4.1 Motivation

The two distances $d\left(Q,\tilde{Q}^{(k)}\right)$, $d\left(S,\tilde{S}^{(k)}\right)$in the second filter of MIR (relation (6-17)) lower its pruning power. That is why the performance of MIR is not as good as that of MIR_X which uses an exclusion condition that does not contain those distances. MIR has one main advantage; it is a standalone method, unlike MIR_X .

On the other hand, although the performance of MIR_X is better than that of MIR, it is completely dependent on the dimensionality reduction technique used. Its application requires adopting a different concept of resolution level for each dimensionality reduction technique, which is not intuitive. Besides, some dimensionality reduction techniques have certain restrictions (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.

6.4.2 The Principle and the Algorithm

The redundancy of the second filter in the case of MIR_X suggests that our multi-resolution algorithm can be applied using two separate filters.

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 Section 6.2. 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 $d\left(Q,\tilde{Q}^{(k)}\right)$, $d\left(S,\tilde{S}^{(k)}\right)$become redundant, so the second filter is overwritten by the usual, more powerful, lower bounding condition $d^R\left(S^{R(k)},Q^{R(k)}\right) > r$.

Notice that the complexity of the modified d^R is $O(2m)$which is the same complexity described in Section 6.2. So this modification does not require any extra cost.

The algorithm we use to apply Tight_MIR is similar to the one described in Section 6.2.4.

6.4.3 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 directly against other dimensionality reduction techniques, because Tight_MIR is a standalone method (unlike MIR_X).

We conducted extensive experiments using datasets of different sizes and dimensions and from different repositories [156], [175], [181], [190].

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, when applying the second filter. Of course the exclusion condition of sequential scanning was also modified in a similar way to avoid this operation. Because this modification is not possible with MIR, the comparison we present here is made between the speed-up of MIR and Tight_MIR compared to sequential scanning. In Figure 6.16 we present the results of four datasets. The results clearly show that Tight_MIR outperforms MIR for all the datasets and for the different values of r. As in the experiments of Sections 6.2 and 6.3, the values of r vary between those which return 1% and 10% of the time series in sequential scanning.

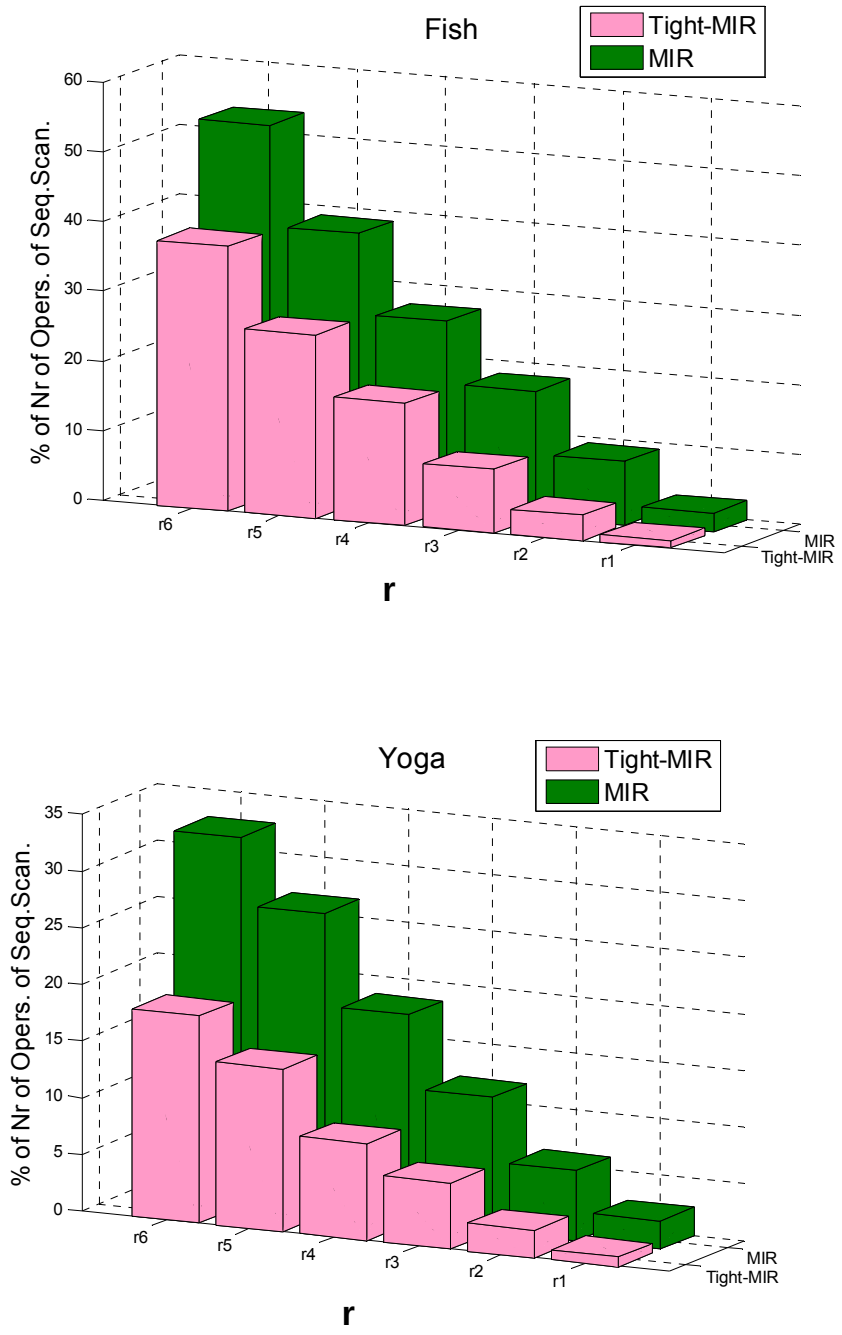
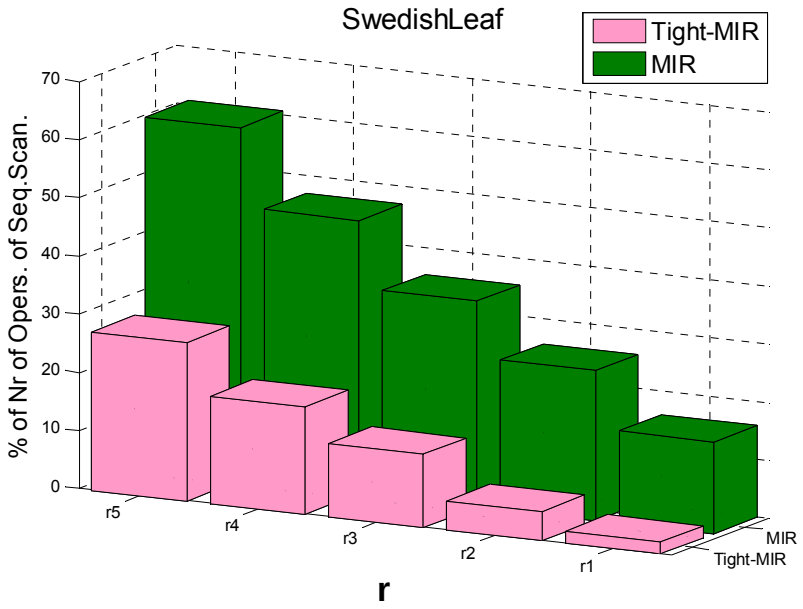


Fig. 6.16. Comparison of the speed-up of MIR and Tight_MIR on four datasets (*Fish*), (*Yoga*), (*SwedishLeaf*), and (*GunPoint*) over sequential scanning.



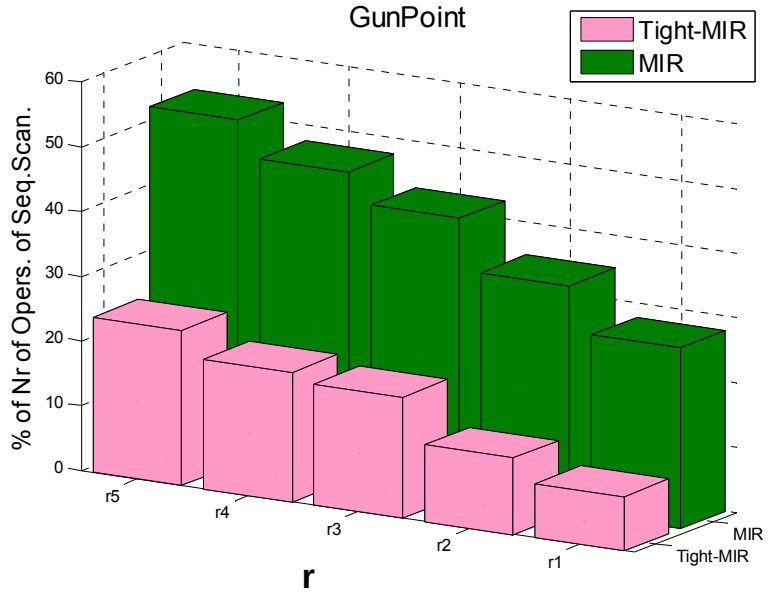


Fig. 6.16. (Continued).

In the second series of experiments we compared MIR_X, with Tight_MIR to show that the two methods give similar results. Figure 6.17 shows some of the results we obtained comparing Tight_MIR with MIR_PAA. The results presented 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.

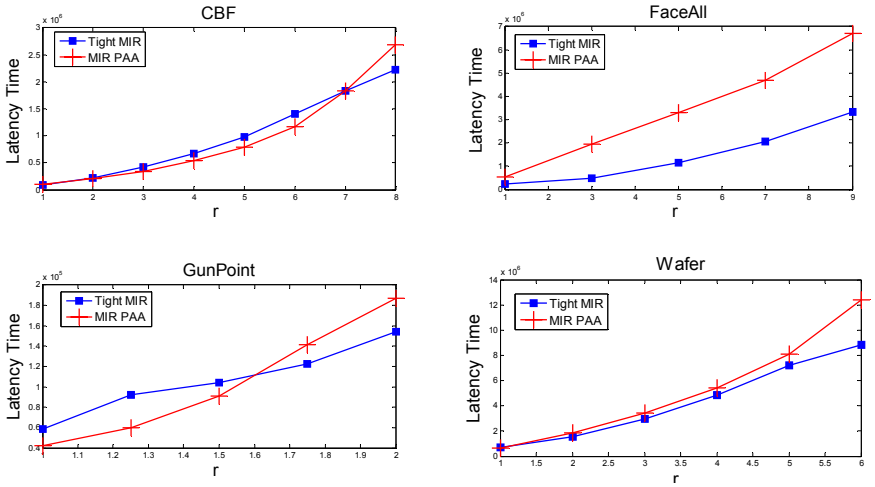


Fig. 6.17. Comparison between MIR_PAA and Tight_MIR on datasets (*CBF*), (*FaceAll*), (*GunPoint*) , and (*Wafer*).

We also conducted other experiments to compare Tight_MIR with PAA, using different datasets and different values of r. We report in Figure 6.18 some of the results we obtained. The results show that Tight_MIR outperforms PAA for all the datasets.

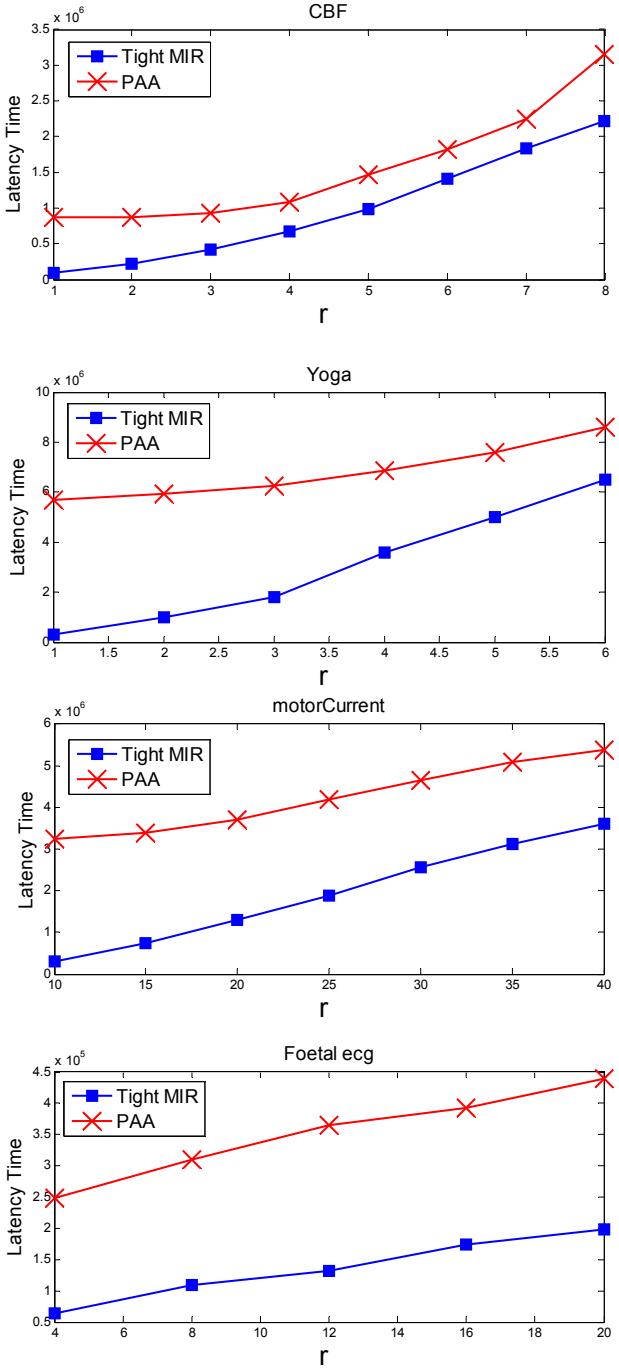


Fig. 6.18. Comparison between PAA and Tight_MIR on datasets (*CBF*), (*Yoga*), (*motorCurrent*), and (*Foetal ecg*)

We also conducted experiments comparing Tight_MIR with SAX on different datasets, and for different values of the alphabet size. The codes we used in the experiments were optimized versions of the original ones, because 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 6.19 the results of several datasets and for different values of r. The results shown here are for alphabet size 3, 10, and 20

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.

It is important to mention that the results of SAX as shown Figure 6.19 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 SAX examined 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 possible number of operations, i.e. the maximum number of distance evaluations.

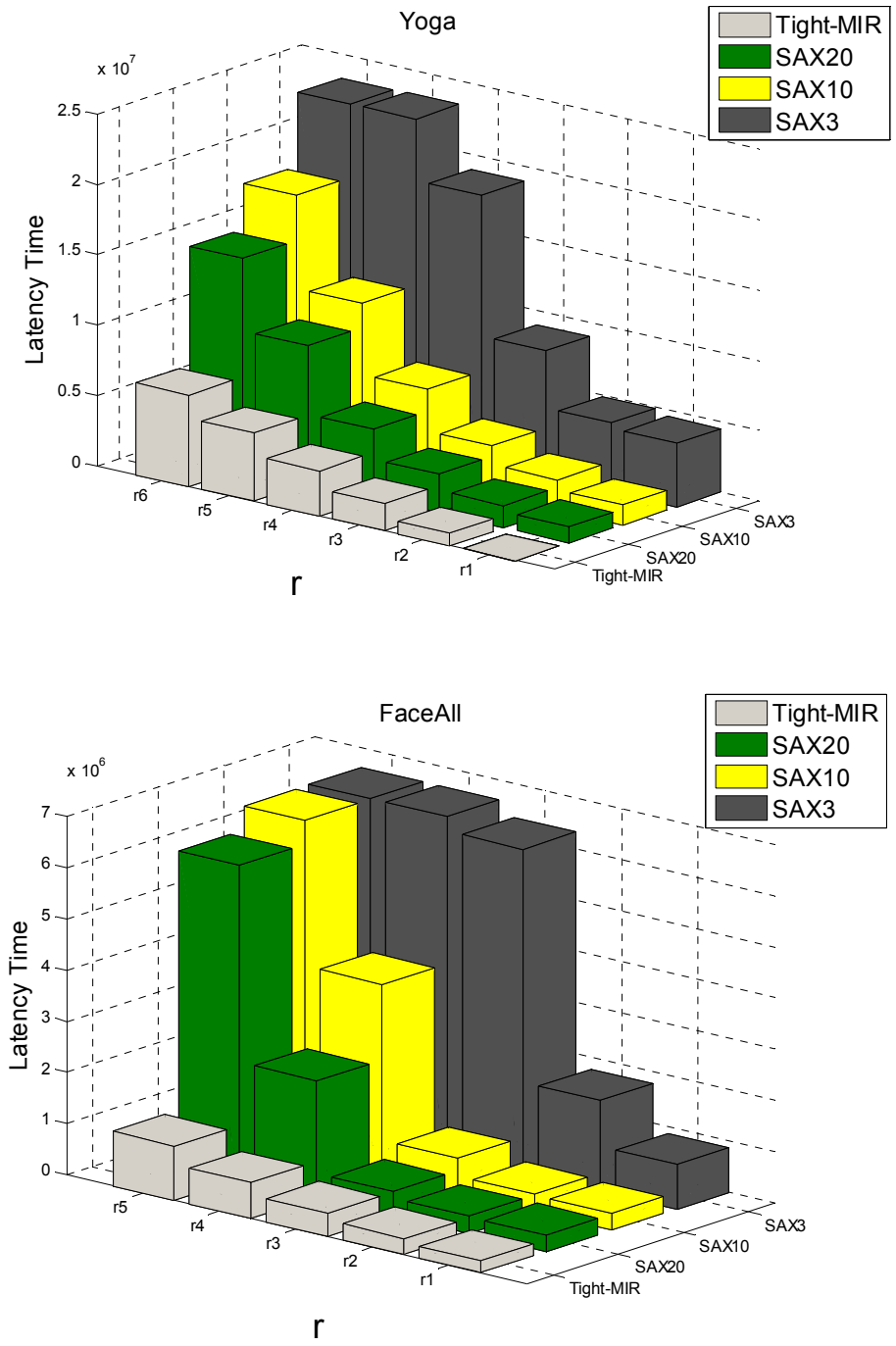
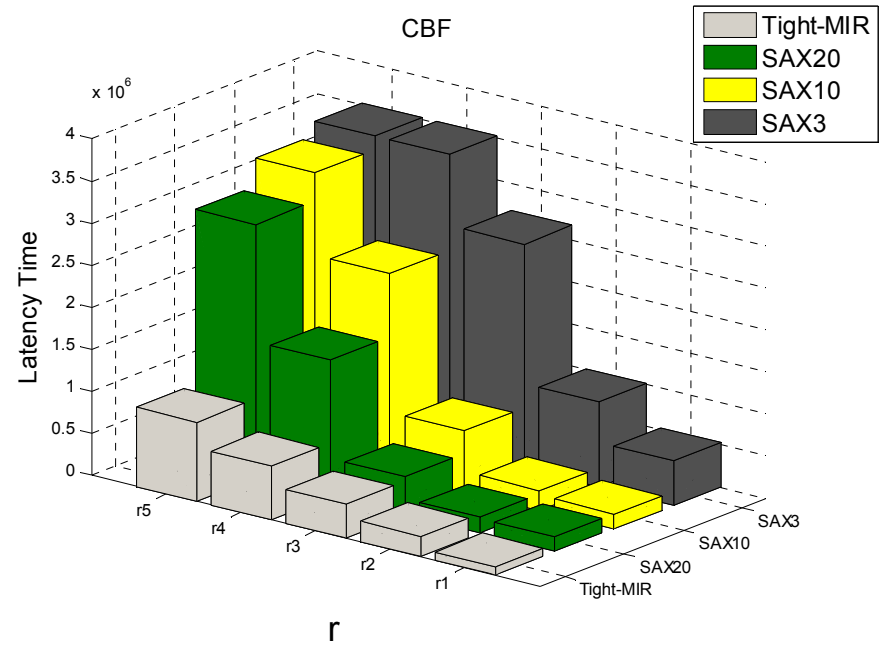


Fig. 6.19. Comparison of the latency time between Tight_MIR and SAX for alphabet size=3, 10, and 20 on datasets (*Yoga*), (*FaceAll*), (*CBF*), and (*motoCurrent*)



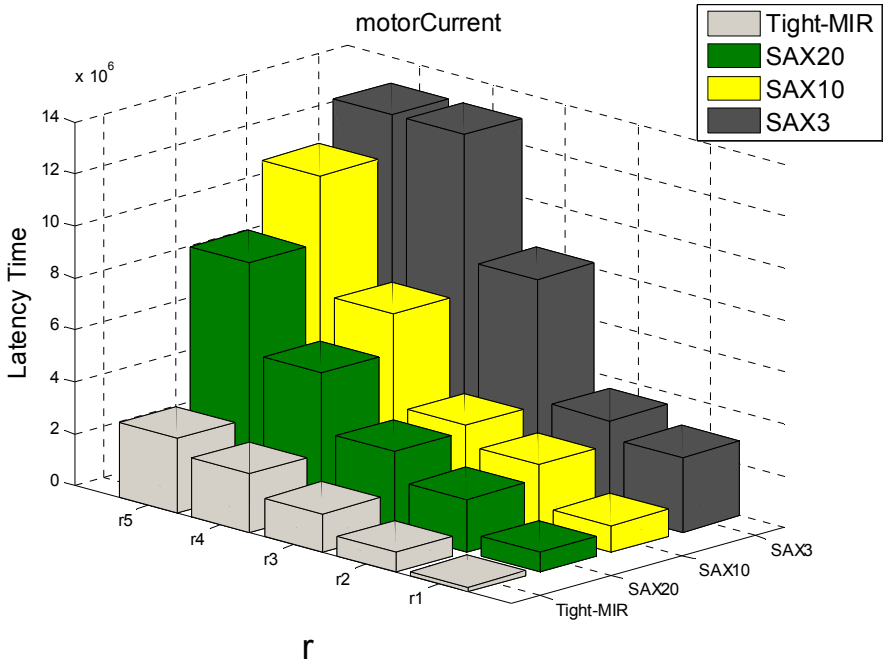


Fig. 6.19. (Continued)

In all the experiments we presented so far the 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 6.2 we present the pruning power of Tight_MIR and SAX with alphabet size=20 (which is the most effective version of SAX) on the datasets presented in Figure 6.19. The results show that the pruning power of Tight_MIR is much more stable than that of SAX20 as r gets larger.

Table 6.2. 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 6.19. The numbers show the percentage of the number of time series that the method excluded to the total number of time series that sequential scanning excludes

	SAX20		**Tight_MIR**	
	$\mathbf{r}_{\mathbf{min}}$	$\mathbf{r}_{\mathbf{max}}$	$\mathbf{r}_{\mathbf{min}}$	$\mathbf{r}_{\mathbf{max}}$
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 lengths of time series. We conducted experiments using datasets of different dimensions and the results showed high stability of performance. We present in Figure 6.20 the results of applying Tight_MIR on dataset (*Wind*) whose length is 12, and dataset (*motorCurrent*) whose length is 1500, compared to sequential scanning which represents the baseline performance.

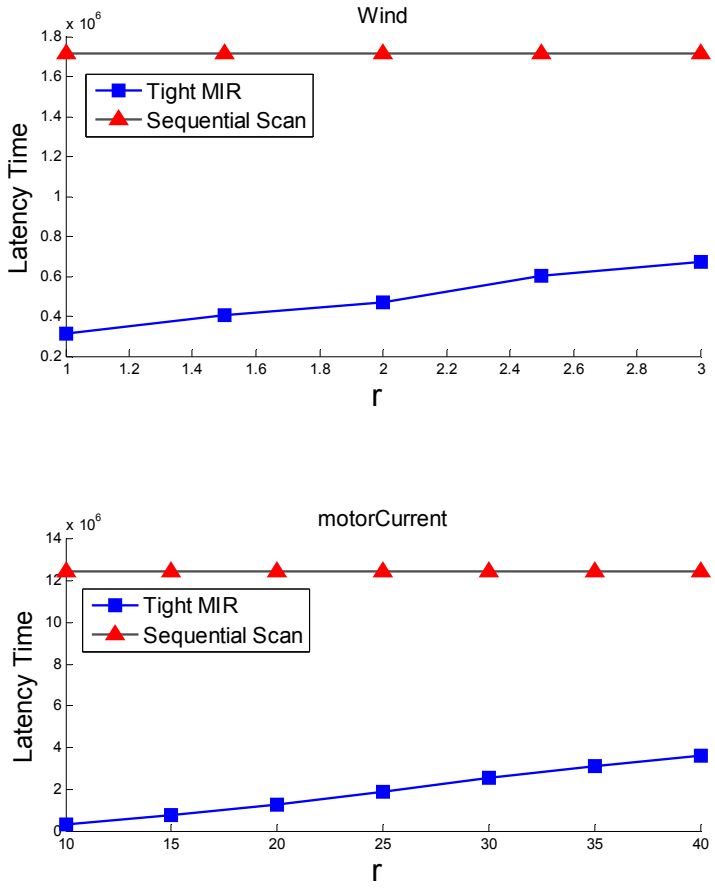


Fig. 6.20. Comparison of the latency time between Tight_MIR and sequential scanning on datasets (*Wind*) and (*motorCurrent*)

6.4.4 Discussion

In this section we presented an improved multi-resolution algorithm of time series retrieval. This new algorithm combines the advantages of the two previously proposed algorithms in this chapter MIR and MIR_X. We conducted extensive experiments comparing the new algorithm with the two other 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 the tested dimensionality reduction techniques both in terms of speed and pruning power.

6.5 What Next?

Our objective in presenting the three multi-resolution algorithms we introduced in this chapter was to show the merits of multi-resolution approaches in economizing distance computations to the lowest degree possible. But we think these are the first steps on a long road of exploiting these approaches, and we think there is still a lot of research to be done on multi-resolution approaches.

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

We also think that multidimensional time series is another direction of future work. The multidimensional nature of these time series data seems to comply with the multi-resolution aspect of our algorithms.

Training the algorithm seems to have many advantages. The step-by-step behavior of the three algorithms we presented in this chapter substantially hinders their performance. The simple training experiment we conducted in Section 6.2.4 shows the potential advantage of training these algorithms. The main difficulty is that we have several parameters that influence the performance of our multi-resolution algorithms. We think a more sophisticated training paradigm using neural networks, which are effective in training algorithms with several parameters, can boost the performance of our algorithms so that the search process can access promising levels directly and terminates immediately when processing indexed data is no longer cost-effective.

Another major direction of future work is to use landmark methods in choosing the points we select at each resolution level. Landmarks models [153] extract points of great importance and identify them as landmark points. This complies with the method of adaptive sampling we presented in Chapter 4 except that in this case the sampling should choose points of great importance. When we were working on our algorithm we thought that choosing the points of each resolution level based on a landmark model could give much better results. The problem with this approach is that the points we select from times series i at a certain resolution level may have different time stamps as time series j at the same resolution level, which is something the Euclidean distance can not deal with. DTW, on the other hand, can deal with time series with different time stamps, but DTW violates the triangle inequality, so it can not be applied with our algorithms, not to mention that it is costly..

We think the answer might come from a new distance metric that can deal with time series of different time stamps (non-uniformly sampled). This distance is called *Time Warp Edit Distance* (TWED) [124]. TWED is defined as follows:

$$\delta_{\lambda,\gamma}\left(A_1^p, B_1^q\right) = min\begin{cases}\delta_{\lambda,\gamma}\left(A_1^{p-1}, B_1^q\right) + \Gamma\left(a'_p \to \Lambda\right)\\ \delta_{\lambda,\gamma}\left(A_1^{p-1}, B_1^{q-1}\right) + \Gamma\left(a'_p \to b'_q\right)\\ \delta_{\lambda,\gamma}\left(A_1^p, B_1^{q-1}\right) + \Gamma\left(\Lambda \to b'_q\right)\end{cases} \quad (6\text{-}18)$$

with

$$\Gamma\left(a'_p \to \Lambda\right) = d\left(a'_p, a'_{p-1}\right) + \lambda$$
$$\Gamma\left(a'_p \to b'_q\right) = d\left(a'_p, b'_q\right) + d\left(a'_{p-1}, b'_{q-1}\right)$$
$$\Gamma\left(\Lambda \to b'_q\right) = d\left(b'_{q-1}, b'_q\right) + \lambda$$

where d is any distance on $\mathbb{R}^{k+1}$. In practice, we choose:

$$d\left(a', b'\right) = d_{LP}\left(a, b\right) + \gamma . d_{Lp}\left(t_a, t_b\right)$$

where γ is a parameter which characterizes the stiffness of the elastic distance $\delta_{\lambda,\gamma}$, and λ any positive constant element in $\mathbb{R}$ that corresponds to a gap penalty.

The recursion is initialized by setting:

$$\delta_{\lambda,\gamma}\left(A_1^0, B_1^0\right) = 0$$
$$\delta_{\lambda,\gamma}\left(A_1^0, B_1^j\right) = \sum_{k=1}^{j} d\left(b'_k, b'_{k-1}\right), j \in \{1,..,q\}$$
$$\delta_{\lambda,\gamma}\left(A_1^i, B_1^0\right) = \sum_{k=1}^{i} d\left(a'_k, a'_{k-1}\right), i \in \{1,..,p\}$$

with $a'_0 = b'_0 = 0$ by convention

As mentioned earlier, this distance is metric and it is applied to time series of different time stamps. So we think it can be an ideal distance to apply with our multi-resolution algorithm using a landmark model. □

Another main direction of future work is to apply our method to other data types where the idea of resolution level is pertinent. The main challenge here is that while the concept of optimal approximation is familiar in time series which are generally numeric data, this concept is unintuitive in other data types. However, the fact that the two filters are separable is beneficial in extending our algorithm to other data types. Although our research on a general model is still in its beginnings, we now have an algorithm that applies our multi-resolution model to symbolic data.

Chapter 7

Conclusion and Future Work

In this dissertation we addressed the problem of similarity search in high-dimensional spaces, mainly from a metric perspective. We showed how the metric model is built and how the similarity search problem is tackled in a metric environment. We also presented the well-known representation methods in the time series literature. We showed the general framework of handling time series data which aims to lower the high-dimensionality of the time series by projecting them onto lower-dimensional spaces and defining a lower bounding distance on these lower-dimensional spaces to guarantee that the algorithm will not produce false dismissals. We also presented different time series dimensionality reduction techniques and showed how they handle this problem using different principles. We showed how symbolic methods are of particular interest because they benefit form the well-known textual and bioinformatics algorithm. We presented one of the most competitive time series symbolic representation methods and showed how by using pre-computed distances this method can substantially speed up the search process.

In the last part of this dissertation we presented multi-resolution methods in multimedia retrieval and showed how these methods provide a new approach to handle the similarity search problem.

7.1 Summary of the Dissertation Contributions

We present in this section the contributions of our work in the order we presented them in this dissertation. These contributions are one minor contribution and two categories of major contributions.

A Minor Contribution: An experimental study which evaluates the impact of dimensionality reduction on a time series classification task. The results we obtained show that compared with other dimensionality reduction techniques and using different similarity measures the performance of the adaptive sampling method remains acceptable even when using a high compression ratio. However, the experiments were conducted on one synthetic dataset. Further experiments on other datasets seem necessary to generalize the results we obtained.

The First Category of Major Contributions: These are the contributions that concern symbolic data and they are grouped in two sub-categories:

- Different versions of extended edit distance metrics. These versions have the advantage of considering global similarity features that the ordinary edit distance does not consider. In addition to versions that use parameters, we proposed a non-parametric version of these metrics. The experiments we conducted on a time series classification task show the superiority of the proposed versions over the edit distance. The versions we presented can be applied to other data types as shown later by another author using one of the versions we proposed.

- A new minimum similarity measure for SAX. The new measure UMD has the same advantages of the original measure but it is tighter and more intuitive. We showed through experiments that our new similarity measure outperforms the original measure on a time series classification task for different values of the alphabet size.

The Second Category of Major Contributions: These are three versions of multi-resolution time series indexing and retrieval algorithms. The main principle of these versions is to reduce the number of query-time distance evaluations by using pre-computed distances which are calculated at indexing-time by projecting the time series on several reduced spaces. These distances are exploited at query-time, using assigned filters based on the triangle inequality, to filter out a large number of time series, which are not answers to the query, without any distance evaluations, or by using lower-cost query-time distance computations. The extensive experiments that we present show the advantages of the proposed multi-resolution versions.

We believe we have met the initial objective that we set at the beginning of the dissertation. Our work included the different aspects of the similarity search problem in high-dimensional spaces, and although the experiments we conducted to test our proposed methods used time series data, most of our work can be extended to other data types.

7.2 Future Work

Although research in multimedia information retrieval has come a long way since its beginnings, we believe there is still a lot to be done. We present in this section some directions of future work in this field of computer science.

While working on this dissertation we realized the need to find different training strategies. In most cases there was an important difference between the performance of the algorithm on testing sets and training sets. We think the reason for this is that, and because of the nature of time series, these training strategies produce over-fitting models. We think other more controllable training strategies should be developed to remedy this problem. □

In multimedia similarity search, the algorithms focus on establishing a system that uses an indexed database. This system is validated on a training set before being deployed. We think future work can use a black box approach to this problem. The algorithm can use several indexing structures and the algorithm is trained on different potential queries, which form a *pattern set*, to learn which indexing structure is best for a certain type of queries. At query-time the algorithm can compare the query with the members of the pattern set and decide, according to some semantics or statistics, which member of the pattern set is closer to the query, then processing the query using the indexing structure that was shown to be best for that member of the pattern set. □

When I was first introduced to time series information retrieval and data mining, the question that confused me was "but where is time in all this?". Now with much confidence I can say "time is not an intrinsic feature of time series indexing and retrieval algorithms". The way time series are presented is not different from the way a function is presented in many applications in numerical analysis where different values of x are given, together with their corresponding y values. Looking at time series in this manner can greatly help see time series representation methods as a strong, may be even the strongest, mathematical tool to handle complex figures. This perspective of representation methods can explain all the new applications of time series data. We think the future of representation methods will focus on this aspect; using time series representation methods to handle different figure-related problems for which applied mathematics could not find efficient answers. □

With the vast number of structures that were proposed to handle the similarity search problem in metric spaces, we strongly believe that future research should focus more on new, intelligent algorithms to exploit these already proposed structures rather than presenting new structures. We give here this example; the performance of the great majority of these structures degrades as the value of r gets larger, yet most structures hardly consider this case. The explanation given for this is that in practice r is usually small. When investigating further, we found out that no paper that we know of defined what is considered large and what is considered small, so in our experiments we tried to consider that any value of r that returns more than 10% of the data objects is large. This may be one step forward, but it is still not enough, because this phenomenon is too complex to be simplified by giving a certain percentage. We know, for instance, that when the query lies in a dense region of data objects then a very "small" value of r may return more than 10% of the data objects. Still, the same query with a relatively "large" value of r may hardly return 1% of the data objects, for the same data set, when the query lies in a sparse region of the data space. Noise also complicates this phenomenon. But the most important thing is that in almost all cases the user does not know at query-time if the value they are assigning to r is "large" or "small", and even this simplified concept which is based on the percentage of the returned data objects would not be helpful because in many cases the user may ignore the size of the database or the distribution of the data objects in the search space. We can see now the consequences of applying an "unintelligent" algorithm to handle this case. Let us say we have a query with a large value of r that returns 90% of the data objects (this is an extreme case to help understand the example). How do

classical algorithms process this query? They project the query on the reduced space, examine the query and try to exclude the data objects, which are not answers to the query, using the distance defined on the reduced space. Of course, in this case this distance fails to exclude at least 90% of the data objects, so the algorithm returns a candidate answer set whose cardinality is at least 90% of the cardinality of the search space set. Ignoring which answers are false alarms, the algorithm has to post-process all this large candidate answer set, using the original distance, just to filter out few false alarms. This scenario is worse than sequential scanning, let alone the overhead storage and indexing-time calculations required. This extreme example shows the need to find more adaptable algorithms that can process queries differently even for the same dataset. □

When time series information retrieval was in its beginning, most papers in this field focused on finding "universal" methods, i.e. methods that can be applied to different datasets. Today, after long research in this field, several very competitive methods have been proposed that it is unlikely to find universal methods which are more competitive than existing ones. On the other hand, there are always more and more new datasets and media. A question arises "is it still worth working on finding universal methods?" We do know that even the most competitive method fails to give the best results for all datasets. Not much research has been done to explore this area of time series information retrieval and data mining. In a classification task, for example, there are many factors that could influence the performance of a time series representation method (the size of the dataset, its nature, the length of the time series, the number of classes, the size of the training set and testing set, the task at hand, etc), and may be there are other factors that we are not aware of. It would be great progress if we could determine all the factors which affect the performance of a certain representation method. But this objective could be too ambitious. Yet another, more feasible goal may be reached. Different existing methods can be applied to a wide variety of datasets. These datasets are then classified according to the most suitable representation method for each dataset. Later whenever a new dataset is introduced, it can be compared with these trained datasets to decide which representation method best suits this dataset. The challenge here is to define a similarity measure between datasets.

References

[1] Achuthsankar, S N.: Computational Biology & Bioinformatics - A gentle Overview, Communications of Computer Society of India, January (2007).

[2] Adistambha, K., Davis, S.J., Ritz, C.H. and Burnett, I.S. :Query Streaming for Multimedia Query by Content from Mobile Devices, in proc. ICSPCS'2007 , Gold Coast, Australia, Dec 17–19 (2007).

[3] Aggarwal, C., Hinneburg, A., and Keim, D. : On the Surprising Behavior of Distance Metrics in High Dimensional Spaces. In Proc. 8th International Conference on Database Theory (ICDT'01). Springer-Verlag, London, UK. (2001).

[4] 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).

[5] 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 (1995).

[6] Allen, E.G., Horvath, M.R.,Kopek, C.V., Lamb, B.T. , Whaples, T.S. and Berry, M.W. :Anomaly Detection Using Non-negative Matrix Factorization, Survey of Text Mining II, Springer London, (2007).

[7] Alonso, C. J., and Rodriguez, J. J.: Boosting Interval Based Literals: Variable Length and Early Classification. In Data Mining in Time Series Databases. World Scientific (2004).

[8] Altschul, S. F., Gish,W., Miller,W., Myers, E.W., and Lipman, D. J.: Basic Local Alignment Search Tool. J Mol Biol, 215(3) (1990).

[9] André-Jönsson, H. & Badal. D.: Using Signature Files for Querying Time-Series Data. In proceedings of Principles of Data Mining and Knowledge Discovery, 1st European Symposium.Trondheim, Norway, Jun 24-27. (1997).

[10] Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y. :An Optimal Algorithm for Approximate Nearest Neighbor Searching, Journal of the ACM 45 (6) (1998).

[11] Asano, T., Ranjan, D., Roos, T., Welzl, E., Widmayer, P.:Space-filling Curves and Their Use in the Design of Geometric Data Structures, Theoretical Computer Science, v.181 n.1, July 15 (1997).

[12] Aßfalg, J., Kriegel, H.-P. , Kröger, P., Kunath, P., Pryakhin,A. and Renz, M. :Similarity Search on Time Series Based on Threshold Queries. In EDBT (2006).

[13] Baeza-Yates R., Cunto W., Manber U., Wu S.: Proximity Matching Using Fixed-queries Trees, Proc. Combinatorial Pattern Matching (CPM'94), LNCS ,(1994).

[14] Barbosa, F.: Similarity-based Retrieval in High Dimensional Data with Recursive Lists of Clusters: A Study Case with Natural Language Dictionaries, in Proc. of the International Conference on Information Management and Engineering (2009).

[15] Bellman, R. : Adaptative Control Processes: A Guided Tour. Princeton University Press, Princeton, NJ, (1961).

[16] Bellman R.: Dynamic Programming. Princeton University Press, Princeton, NJ (1957).

[17] Belussi, A. , and Faloutsos, C.: Estimating the Selectivity of Spatial Queries Using the 'Correlation' Fractal Dimension, Proceedings of the 21st International Conference on Very Large Databases (VLDB'95), Zurich, Switzerland, (1995).

[18] Berchtold S., Böhm C., Kriegel H.-P.: Improving the Query Performance of High-dimensional Index Structures Using Bulk-Load Operations, 6th. Int. Conf. on Extending Database Technology, Valencia, Spain (1998).

[19] Berchtold S., Böhm C., Kriegel H.-P.: The Pyramid-Technique: Towards indexing beyond the Curse of Dimensionality', Proc. ACM SIGMOD Int. Conf. on Management of Data, Seattle (1998).

[20] Berchtold, S., Keim, D. A., and Kriegel, H.-R: A Cost Model for Nearest Neighbor Search in High-dimensional Data Space. In Proceedings of the 16th ACM Symposium on Principles of Database Systems (PODS 1997), Tucson, Arizona, USA, May 12-14 (1997).

[21] Bergeron, R. D., and Foulks,A. :Interactive Out-of-Core Visualization of Multiresolution Time Series Data, Numerical Modeling of Space plasma flows : ASTRONUM-2006, proceedings of the 1st IGPP-CalSpace International Conference, Palm Springs, California, March (2006).

[22] Berndt, D. and Clifford, J.: Using Dynamic Time Warping to Find Patterns in Time Series. In Proc. AAAI Workshop on Knowledge Discovery in Databases (1994).

[23] Böhm, C. , Berchtold, S., and Keim, D. A. :Searching in High-dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases. ACM Computing Surveys (2001).

[24] Boreczky, J. S. and Rowe, L. A. :Comparison of Video Shot Boundary Detection Techniques. In Proc. 8th Int. Symp. on Storage and Retrieval for Image and Video Data-bases (1996).

[25] Boyd, S.: TREND: A System for Generating Intelligent Descriptions of Time-series Data. In Proceedings of the IEEE International Conference on Intelligent Processing Systems (ICIPS'98). IEEE Press, (1998).

[26] Bramer, M.: Principles of Data Mining . Springer (2007).

[27] Brin, S.: Near Neighbor Search in Large Metric Spaces. In Proc. 21st Conference on Very Large Databases ,VLDB'95 (1995).

[28]Burkhard, W. A. and Keller, R. M.: Some Approaches to Best-match File Searching. Communications of the ACM (1973).

[29] Bustos, B., Navarro, G., and E. Chavez, E.: Pivot Selection Techniques for Proximity Searching in Metric Spaces. Pattern Recognittion Letters (2003).

[30] Bustos, B., and Skopal, T. :Beyond the Metric Space Model", SIGSPATIAL Special volume 2 Issue 2, July (2010).

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

[32] Camerra, A., Palpanas, T., Shieh, J., and Keogh, E. (2010): *iSAX* 2.0: Indexing and Mining One Billion Time Series. ICDM (2010).

[33] Castelli, V. and Lawrence Bergman, L.: Image Databases: Search and Retrieval of Digital Imagery. John Wiley & Sons (2001).

[34] Castro, N., and Azevedo, P. :Multiresolution Motif Discovery in Time Series, Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA. SIAM (2010).

[35] Chan, K.P., and Fu, A.W-C.: Efficient Time Series Matching by Wavelets. In Proc. 15th. Int. Conf. on Data Engineering (1999).

[36] Chan, K.P., Fu, A and Yu, C. :Haar Wavelets for Efficient Similarity Search of Time-series: with and without Time Warping. IEEE Transactions on Knowledge and Data Engineering (2002).

[37] Chávez, E., and Navarro, Baeza-Yates, R., and Marroquín, J. : Proximity Searching in Metric Spaces. ACM Computing Surveys (2001).

[38] Chavez, E., Navarro, G., Baeza-Yates, R. A., and Marroquin, J. L.: Searching in Metric Spaces. ACM Computing Surveys (2001)

[39] Chávez, E., and Navarro, G. : Fundamentals of the Problem , SIGSPATIAL Special volume 2 Issue 2, July (2010).

[40] Chávez, E., and Navarro, G. : Metric Databases, Encyclopedia of Database Technologies and Applications. Idea Group (2006).

[41] Chen, H., Li, J. and Mohapatra, P. :RACE: Time Series Compression with Rate Adaptivity and Error Bound for Sensor Networks. In Proc. of IEEE MASS (2004).

[42] Chen, L., Ng, R. :On the Marriage of Lp-Norm and Edit Distance, In Proceedings of 30th International Conference on Very Large Data Base, Toronto, Canada, August (2004).

[43] Chen, L., Özsu, M. T. and Oria, V. : Robust and Fast Similarity Search for Moving Object Trajectories. In SIGMOD Conference (2005).

[44] Chen, L.: Similarity Search Over Time Series and Trajectory Data, PhD thesis, University of Waterloo, Canada (2005)

[45] Chen, Q., Chen, L., Lian, X., Liu, Y. and Yu, J. X. : Indexable PLA for Efficient Similarity Search. In VLDB (2007).

[46] Chen, Y., Nascimento, M. A., Ooi, B. C., and Tung, A. K. H. : SpADe: On Shape-based Pattern Detection in Streaming Time Series. In ICDE (2007).

[47] Chu, S., Keogh, E., Hart, D., Pazzani, M.: Iterative deepening dynamic time warping for time series. In Proc 2nd SIAM International Conference on Data Mining (2002).

[48] Ciaccia, P., Patella, M. :The Many Facets of Approximate Similarity Search, in Proceedings of the First International Workshop on Similarity Search and Applications (SISAP 2008), Cancun, Mexico, IEEE Computer Society (2008).

[49] Ciaccia, P., Patella, M., and Zezula, P.: A Cost Model for Similarity Queries in Metric Spaces. In Proceedings of the 17th ACM Symposium on Principles of Database Systems (PODS 1998), Seattle, Washington, USA, June 1-3 (1998).

[50] Ciaccia, P., Patella, M., and Zezula, P., M-Tree: An Efficient Access Method for Similarity Search in Metric Spaces. In Proceedings of the 23rd Conference on Very large Database, VLDB'97 (1997).

[51] CLCBio : Bioinformatics explained: BLAST versus Smith-Waterman, Gustav Wieds (2007)

[52] Cormen, T.H., Leiserson, C.E., and Rivest, R.L.: Introduction to Algorithms.The MIT Electrical Engineering and Computer Science Series. MIT Press/McGraw Hill, (1990).

[53] Das, G., Lin, K.-I., Mannila, H., Renganathan G. and Smyth, P.: Rule Discovery from Time Series Database: In Proc. of the 4th Intl. Conference on Knowledge Discovery and Data Mining KDD (1998).

[54] Daw, C. S., Finney, C. E. A. and Tracy, E. R. : Symbolic Analysis of Experimental Data. Review of Scientific Instruments (2002).

[55] DeVore, R., Jawerth, B. and Lucier, B.: Image Compression Through Wavelet Transform Coding. IEEE Transactions on Information Theory (1992).

[56] Dickerson, K. B., and Ventura, D.: Music Recommendation and Query-by-content Using Self-organizing Maps, Neural Networks, IEEE - INNS - ENNS International Joint Conference (2009).

[57] 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).

[58] Dohnal, V.: Indexing Structures for Searching in Metric Spaces.PhD thesis, Masaryk University, Faculty of Informatics (2004).

[59] Dohnal, V., Gennaro, C., Savino, P., and Zezula, P.: Similarity Join in Metric Spaces. In Advances in Information Retrieval, 25th European Conference on IR Research, ECIR 2003. Pisa, Italy,(2003)

[60]Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis, A Wiley-Interscience Publication, New York: Wiley (1973)

[61] Dumais, S. T. : Latent Semantic Indexing (LSI) and TREC-2., In: D. Harman (Ed.), The Second Text REtrieval Conference (TREC2), National Institute of Standards and Technology Special Publication (1994).

[62] Eruhimov, V., Martyanov, V., Raulefs, P., and Tuv, E.: Combining Unsupervised and Supervised Approaches to Feature Selection for Multivariate Signal Compression., Intelligent Data Engineering and Automated Learning, Springer, Berlin, Germany (2006).

[63] Faloutsos, C, Barber, R., Flickner, M., Hafner, J., Niblack, W., Petkovic,D., and Equitz, W.: Efficient and Effective Querying by Image Content. Journal of Intelligent Information Systems, Kluwer Academic Publishers (1994).

[64] Faloutsos C.: Gray Codes for Partial Match and Range Queries, IEEE Trans. on Software Engineering (1988).

[65] Faloutsos, C., Ranganathan, M., and Manolopoulos, Y.: Fast Subsequence Matching in Time-series Databases. In Proc. ACM SIGMOD Conf., Minneapolis (1994).

[66] Ferhatosmanoglu, H., Tuncel, E., Agrawal, D., and Abbadi, A. E.: Approximate Nearest Neighbor Searching in Multimedia Databases. In Proceedings of the 17th International Conference on Data Engineering (ICDE 2001), Heidelberg,Germany, April 2-6, 2001. IEEE Computer Society (2001).

[67] Figueras i Ventura R. M., Frossard P. and Vandergheynst P.: Evolutionary Multiresolution Matching Pursuit and Its Relations with the Human Visual System, in Proceedings of the European Signal Processing Conference, Toulouse, France, September (2002).

[68] Frank, A. U. and Timpf, S.: Multiple Representations for Cartographical Objects in a Multi-scale Tree - An Intelligent Graphical Zoom. Computers and Graphics (1994).

[69] Frentzos, E., Gratsias, K., and Theodoridis, Y.: Index-based Most Similar Trajectory Search. In ICDE (2007).

[70] Gansterer, W.N., Janecek, A.G.K., and Neumayer, R.: Spam Filtering Based on Latent Semantic Indexing. Survey of Text Mining II - Clustering, Classification, and Retrieval, volume 2, Springer (2008).

[71] Gennaro, C, Savino, P., and Zezula, P.: Similarity Search in Metric Databases Through Hashing. In Proceedings of the 3rd ACM Multimedia 2001 Workshop on Multimedia Information Retrieval (MIR 2001), Ottawa, Ontario, Canada, October 5 (2001).

[72] Geurts, P.: Pattern Extraction for Time Series Classification. Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'01), Springer (2001).

[73] Guo,AY., and Siegelmann, H.: Time-warped Longest Common Subsequence Algorithm for Music Retrieval, in Proc. ISMIR (2004).

[74] Guttman A. 'R-trees: A Dynamic Index Structure for Spatial Searching', Proc. ACM SIGMOD Int. Conf. on Management of Data, Boston, MA (1984).

[75] Han, J., and Kamber, M. : Data Mining:Concepts and Techniques. Morgan Kaufmann Publishers, CA (2005).

[76] Hao, M., Dayal, U., Keim, D. A., Schreck, T.: Multi-resolution Techniques for Visual Exploration of Large Time-Series Data. Proc. of Eurographics/IEEE-VGTC Symposium on Visualization (2007).

[77] Hébrail G., Hugueney B.: Symbolic Representation of Long Time Series, Conference on Applied Statistical Models and Data Analysis (ASMDA'2001), Compiègne, Juin (2001).

[78] Hébrail G., Hugueney B.: Symbolic Representation of Long Time Series, Workshop on Symbolic Data Analysis (PKDD'2000), Lyon, Septembre (2000).

[79] Hetland, M. L.: The Basic Principles of Metric Indexing, Swarm Intelligence for Multi-objective Problems in Data Mining, SCI 242, Springer (2009).

[80]Hiremath, P. S., and Pujari, J. :Content Based Image Retrieval based on Color, Texture and Shape features using Image and Its Complement". International Journal of Computer Science and Security, 1(4), (2007).

[81] Howland, P., and Park, H.: Cluster-preserving Dimension Reduction Methods for Efficient Classification of Text Data. In M. W. Berry (Ed.), Survey of text mining: Clustering, classification, and retrieval, New York: Springer-Verlag (2004).

[82] Huang, Y. and Yu, P. S.: Adaptive Query Processing for Time-Series Data. In proceedings of the 5th Int'l Conference on Knowledge Discovery and Data Mining. San Diego, CA, Aug 15-18 (1999).

[83] Indyk, P.: Algorithmic Applications of Low-distortion Geometric Embeddings. In Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (2001).

[84] Jacobs, C. E., Finkelstein, A., and Salesin, D. H.: Fast Multiresolution Image Querying. In Proceedings of SIGGRAPH 95, ACM, New York (1995).

[85] Jahromi, M.Z., Parvinnia, E., John, R.: A Method of Learning Weighted Similarity Function to Improve the Performance of Nearest Neighbor, Inf. Sci. 179 (2009).

[86] Kadous, M. W.: Learning Comprehensible Descriptions of Multivariate Time Series. In Proc. of the 16th International Machine Learning Conference (1999).

[87] Kalantari, I., McDonald, G.: A Data Structure and an Algorithm for the Nearest Point Problem, IEEE Transactions on Software Engineering, vol. 9, no. 5, September (1983).

[88] Kamel I., Faloutsos C.: On Packing R-trees, CIKM (1993).

[89] Kawagoe, K., and Ueda, T.: A Similarity Search Method of Time Series Data with Combination of Fourier and Wavelet Transforms. In TIME (2002).

[90] Keikha, M., Razavian, N.S., and Oroumchian, F.: Document Representation and Quality Of Text: An Analysis, Survey of Text Mining: Clustering, Classification and Retrieval, Second Edition, Springer (2007).

[91] Keogh, E. : A Tutorial on Indexing and Mining Time Series Data , http://www.cs.ucr.edu/~eamonn/tutorials.html

[92] Keogh, E,. Chakrabarti, K,. Pazzani, M. and Mehrotra, S.: Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. J. of Know. and Inform. Sys. (2000).

[93] Keogh, E,. Chakrabarti, K,. Pazzani, M. and Mehrotra, S.: Locally Adaptive Dimensionality Reduction for Similarity Search in Large Time Series Databases. SIGMOD (2001).

[94] Keogh, E.: Exact Indexing of Dynamic Time Warping. In proceedings of the 28th International Conference on Very Large Data Bases. Hong Kong, Aug 20-23 (2002).

[95] Keogh, E., and Kasetty, S. : On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. July 23 - 26, 2002. Edmonton, Alberta, Canada (2002).

[96] Keogh, E., Lin, J., and Fu, A.: HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In Proc. of the 5th IEEE International Conference on Data Mining (ICDM 2005), Houston, Texas, Nov 27-30 (2005).

[97] Keogh, E., and Pazzani, M.: An Enhanced Representation of Time Series Which Allows Fast and Accurate Classification, Clustering, and Relevance Feedback. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98), AAAI Press (1998).

[98] Kim, J., and Warnow, T.: Tutorial on Phylogenetic Tree Estimation. In Intelligent Systems for Molecular Biology, Heidelberg (1999).

[99] Kim, S,. Park, S., & Chu, W.: An Index-based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases. In Proc 17th International Conference on Data Engineering (2001).

[100] Kogan, J.: Hybrid Clustering of Large Text Data. In Proceedings of the IEEE 21st International Conference on Advanced Information Networking and Applications (AINA-07), IEEE Computer Society Press, (2007).

[101] Kohonen, T.; Self-Organization and Associative Memory. Springer (1984).

[102] Kolesnikov, A., and Franti, P.: Reduced-search Dynamic Programming for Approximation of Polygonal Curves. Pattern Recognition Letters (2003).

[103] Kollios, G., Vlachos, M. and Gunopulos, G.: Discovering similar multidimensional trajectories. In Proc 18th International Conference on Data Engineering (2002).

[104] Korn, F., Jagadish, H and Faloutsos. C.: Efficiently Supporting ad hoc Queries in Large Datasets of Time Sequences. Proceedings of SIGMOD '97, Tucson, AZ (1997).

[105] Kom, F. and Muthukrishnan, S. M.: Influence Sets Based on Reverse Nearest Neighbor Queries. In Proceedings of the ACM International Conferenceon Management of Data (SIGMOD 2000), Dallas, Texas, USA, May 16-18, (2000).

[106] Kurtz, S. : Lecture Notes for Foundations of Sequence Analysis (2001).

[107] Larose, D.T.: Discovering Knowledge in Data: An Introduction to Data Mining. New York, Wiley (2005).

[108] Larsen, R. J. and Marx, M. L.: An Introduction to Mathematical Statistics and Its Applications. Prentice Hall, Englewood, Cliffs, N.J. 2nd Edition. (1986)

[109] Lee, J.A. and M. Verleysen,M. : Nonlinear Dimensionality Reduction. Springer (2007).

[110] Levenshtein, V. I.: Binary Codes Capable of Correcting Spurious Insertions and Deletions of Ones. Problems of Information Transmission , 1:8-17. Kluwer Academic Publishers (1965).

[111] Lin, J., Keogh, E., Lonardi, S., and Chiu, B. Y.: A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. DMKD (2003).

[112] Lin, J., Keogh, E., Wei, L., and Lonardi, S.: Experiencing SAX: a Novel Symbolic Representation of Time Series. DMKD Journal (2007).

[113] Lin, J., Vlachos, M., Gunopulos, D., Keogh, E.: Multi-Resolution Time Series Clustering and Application to Images, Multimedia Data Mining and Knowledge Discovery, Springer (2007)

[114] Lin, J., Vlachos, M., Keogh, E., and Gunopulos, D.: A MPAA-based Iterative Clustering Algorithm Augmented by Nearest Neighbors Search for Time-series Data Streams. Proceedings of the 9th Pacic-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'05), Springer (2005).

[115] Li, Z., Wang, L., and Zhang, K.: Algorithmic Approaches for Genome Rearrangement: A Review," IEEE Trans. Systems, Man, and Cybernetics, vol. 36 (2006).

[116] Lomet D., Salzberg B.: The hB-tree: A Multiattribute Indexing Method with Good Guaranteed Performance', ACM Trans. on Data Base Systems 15(4) (1990).

[117] Makhanov, S. S., Anotaipaiboon, W.: Advanced Numerical Methods to Optimize Cutting Operations of Five Axis Milling Machines, Springer (2007)

[118] Mamou, J., Mass, Y., Shmueli-Sheuer, M. , and Sznajder, B.: Query Language for Multimedia Content. In Procedding of the Multimedia Information Retrieval workshop held in conjunction with the 30th Annual International ACM SIGIR Conference 27 July 2007, Amsterdam (2007).

[119] Mandl, T.: Learning Similarity Functions in Information Retrieval. In EUFIT (1998).

[120] Manganaris, S.: Supervised Classification with Temporal Data. PhD thesis, Computer Science Department, School of Engineering, Vanderbilt University (1997)..

[121] Marteau, P.F., and Gibet, S.: Adaptive Sampling of Motion Trajectories for Discrete Task-Based Analysis and Synthesis of Gesture. In Lecture Notes in Computer Science, Volume 3881/2006, 224-235,, In Proc. of Int. Gesture Workshop, Vannes, France (2005).

[122] Marteau, P.F., Muhammad Fuad, M.M , and Ménier, G. : Echantillonnage Adaptatif et Classification Supervisée de Séries Temporelles. EGC 2006, 6èmes Journées, Extraction et Gestion de Connaissances, Atelier 9 : Fouille de données temporelles. Jan 17 2006, Lille, France (2006).

[123] Marteau, P.F., Ménier, G., Adaptive Multiresolution and Dedicated Elastic Matching in Linear Time Complexity for Time Series Data Mining, Sixth International conference on Intelligent Systems Design and Applications (ISDA 2006), Jinan Shandong, China, 16-18 October (2006).

[124] Marteau, P.F.:Time Warp Edit Distance with Stiffness Adjustment for Time Series Matching, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 2 (2009).

[125] Mason, J. C., and Handscomb, D.: Chebyshev Polynomials. Chapman & Hall (2003).

[126] Megalooikonomou, C.: Multiresolution Symbolic Representation of Time Series. In proceedings of the 21st IEEE International Conference on Data Engineering (ICDE). Tokyo, Japan. (2005).

[127] Meratnia, N., By, R.: Spatiotemporal Compression Techniques for Moving Point Objects. In: Proceedings of EDBT (2004).

[128] Muhammad Fuad, M.M, Marteau, P.F.: Enhancing the Symbolic Aggregate Approximation Method Using Updated Lookup Tables. 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems – KES

2010 , Lecture Notes in Computer Science, 2010, Volume 6276/2010, 8-10 September 2010, Cardiff, Wales, UK (2010)

[129] Muhammad Fuad, M.M, Marteau, P.F.: Extending the Edit Distance Using Frequencies of Common Characters. 19th International Conference on Database and Expert Systems Applications - DEXA'08, Lecture Notes in Computer Science, 2008, Volume 5181/2008, 1-5 September,2008, Turin, Italy (2008).

[130] Muhammad Fuad, M.M, Marteau, P.F.: Fast Retrieval of Time Series by Combining a Multiresolution Filter with a Representation Technique. The International Conference on Advanced Data Mining and Applications–ADMA2010 , Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence, 2010, Volume 6440/2010, 19-21 November 2010, ChongQing, China (2010).

[131] Muhammad Fuad, M.M, Marteau, P.F.: Multi-resolution Approach to Time Series Retrieval", Fourteenth International Database Engineering and Applications Symposium– IDEAS 2010 , 16-18 August, 2010, Montreal, QC, CANADA (2010).

[132] 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 , 22-24 September 2010, Carnegie Mellon University, Pittsburgh, PA, USA (2010).

[133] Muhammad Fuad, M.M, Marteau, P.F.: The Extended Edit Distance Metric. Sixth International Workshop on Content-Based Multimedia Indexing (CBMI 2008) 18-20th June, 2008, London, UK (2008).

[134] Muhammad Fuad, M.M, Marteau, P.F.: The Multi-resolution Extended Edit Distance. Third International ICST Conference on Scalable Information Systems, Infoscale, 2008, ACM Digital Library, June 4-6, 2008. Vico Equense, Italy (2008).

[135] Muhammad Fuad, M.M, Marteau, P.F.: The Prolonged Multi-Resolution Edit Distance. Dutch-Belgian Information Retrieval workshop, DIR-2008,April 14-15, 2008, Maastricht, the Netherlands (2008).

[136] Muhammad Fuad, M.M, Marteau, P.F.: Towards a Faster Symbolic Aggregate Approximation Method", 5th International Conference on Software and Data Technologies – ICSOFT 2010, 22-24 July 2010, Athens, Greece (2010).

[137] Muhammad Fuad, M.M, Marteau, P.F. : Une Distance d'Édition Etendue Multi Résolution. Atelier à EGC 2008, INRIA, Sophia Antipolis, France (2008).

[138] Micó, L., Oncina, J., and Vidal, E.: A New Version of the Nearest-neighbor Approximating and Eliminating Search (AESA) with Linear Preprocessing-time and Memory Requirements. Pattern Recognition Letters (1994).

[139] Moens, M.F.: Information Extraction: Algorithms and Prospects in a Retrieval Context; The Information Retrieval Series , Vol. 21, Springer (2006).

[140] Mörchen, F.: Time Series Knowledge Mining, PhD thesis, Philipps-University Marburg, Germany, Görich & Weiershäuser, Marburg, Germany (2006)

[141] 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 (2001).

[142] Morse M. D., and Patel, J. M.: An Efficient and Accurate Method for Evaluating Time Series Similarity. In SIGMOD Conference (2007).

[143] Movellan, J.R.: Primer on the Discrete Fourier Transform, Tutorial,Machine Perception Labratory (2009) http://mplab.ucsd.edu/wordpress/ .

[144] Nanopoulos, A., Alcock, R., and Manolopoulos, Y. : Feature-based Classification of Time-Series Data. Information Processing and Technology (2001).

[145]Navarro, G.: Searching in Metric Spaces by Spatial Approximation In Pro. 6th South American Symposium on String Processing and Information Retrieval (SPIRE'99). IEEE CS Press (1999).

[146] Niblack, W., Barber, R., Equitz, W., Flickner, M., Glasman, E., Petkovic, D., Yanker, P., Faloutsos, C., and Taubin, G.: The QBIC Project: Querying Images by Content Using Color, Texture and Shape. In Proc. 5th Int. Symp. on Storage and Retrieval for Image and Video Databases (1993).

[147] Oppenheim, A.V., and Schafer, R.W.: Discrete-Time Signal Processing 2nd Edition, Prentice Hall: Upper Saddle River, NJ (1999).

[148] Orenstein, J.: A Comparison of Spatial Query Processing Techniques for Native and Parameter Spaces. Proc. ACM SIGMOD Int. Conf. on Management of Data (1990).

[149] Papadopoulos, A. N. and Manolopoulos, Y.: Nearest Neighbor Search: A Database Perspective. Springer-Verlag, New York (2005).

[150] Parent, C., Spaccapietra, S., and Zimanyi, E.: The MurMur Project: Modeling and Querying Multi-representation Spatio-temporal Databases. In Information Systems, volume 31 (2006).

[151] Patella, M. and Ciaccia, P.: Approximate Similarity Search: A Multi-faceted Problem. Journal of Discrete Algorithms 7 (1) (2009).

[152] Perez, J. C., and Vidal, E.: Optimum Polygonal Approximation of Digitized Curves. Pattern Recognition Letters (1994).

[153] Perng, S., Wang, H., Zhang, S., and Parker, D.S.: Landmarks: A New Model for Similarity-Based Pattern Querying in Time Series Databases. In Proc. of ICDE (2000).

[154] Pestov, V.: Intrinsic Dimensionality. SIGSPATIAL Special volume 2 Issue 2, July (2010).

[155] Popivanov, I., and Miller, R. J.: Similarity Search over Time Series Data Using Wavelets. ICDE (2002).

[156] http://povinelli.eece.mu.edu/

[157] Proietti, G., and Faloutsos, C.: Analysis of Range Queries and Self-Spatial Join Queries on Real Region Datasets Stored using an R-tree", IEEE Transactions on Knowledge and Data Engineering, Vol.12, No.5 (2000).

[158] Rabiner, L., and Juang, B.: Fundamentals of speech recognition. Englewood Cliffs, N.J, Prentice Hall (1993)..

[159] Ramella, G., Sanniti di Baja, G.: Multiresolution Histogram Analysis for Color Reduction, Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications - 15th Iberoamerican Congress on Pattern Recognition, CIARP 2010, Sao Paulo, Brazil, November 8-11, 2010. Proceedings. Lecture Notes in Computer Science 6419 Springer (2010).

[160] Ravi Kanth, K. V., Agrawal, D., and Singh, A.: Dimensionality Reduction for Similarity Searching in Dynamic Databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data (1998).

[161] Reinert, G., Schbath, S. and Waterman, M. S. Probabilistic and Statistical Properties of Words: An Overview. Journal of Computational. Biology. Vol. 7 (2000).

[162] Sakoe, H., and Chiba, S.: Dynamic Programming Algorithm Optimization for Spoken Word Recognition. IEEE Trans. Acoustics, Speech and Signal Processing, ASSP-26(1), (1978).

[163] Salomon, D., A Concise Introduction to Data Compression. Undergraduate Topics in Computer Science. Springer(2008).

[164] Salomon , D . , Motta , G . : Handbook of Data Compression, Fifth Edition, Springer (2010)

[165] Saito, N.:Local feature extraction and its application using a library of bases. PhD thesis, Yale University (1994)

[166] Sankoff D, El-Mabrouk N: Genome Rearrangement. In Current Topics in Computational Biology. Edited by Jiang T, Smith T, Xu Y, Zhang M, Xu Y, Zhang M. Cambridge: MIT Press (2001).

[167] 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).

[168] Scott, D.W., and Thompson, J.R.: Probability Density Estimation in Higher dimensions. Proceedings of the Fifteenth Symposium on the Interface. Elsevier Science Publishers, B.V., North Holland (1983).

[169] Senellart, P., and Blondel, V.D.: Automatic Discovery of Similar Words. A Comprehensive Survey of Text Mining. Springer-Verlag (2003).

[170] Setubal, J. C. and Meidanis, J.: Introduction to Computational Molecular Biology. PWS Publishing Co., Boston (1997).

[171] Shatkay, H.,: The Fourier Transform - a Primer. Technical Report CS-95-37, Department of Computer Science, Brown University (1995).

[172] Shatkay, H., and Zdonik, S.B.: Approximate Queries and Representations for Large Data Sequences. In Proc. 12th Int. Conf. on Data Engineering (1996).

[173] Shieh, J. and Keogh, E.: *iSAX*: Disk-Aware Mining and Indexing of Massive Time Series Datasets. Data Mining and Knowledge Discovery (2009).

[174] Shieh, J. and Keogh, E.: *iSAX*: Indexing and Mining Terabyte Sized Time Series. In Proceeding of the 14th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24 – 27 (2008).

[175] SISTA's Identification Database
http://www.esat.kuleuven.ac.be/~tokka/daisydata.html

[176] Skopal, T., and Bustos, B.: On Non-metric Similarity Search Problems in Complex Domains. ACM Computing Surveys, to appear.44(3) (2012).

[177] Smith, T. F. and Waterman, M.S.: Identification of Common Molecular Subsequences. J Mol Biol, 147(1), (1981).

[178] Sripada, S. G., Reiter, E., and Hunter, J.: Generating English Summaries of Time Series Data Using the Gricean Maxims. Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'03 (2003).

[179] Stanoi, I., Agrawal, D., and Abbadi, A. E.: Reverse Nearest Neighbour Queries for Dynamic Databases. In Proceedings of the ACM SIGMOD Workshop on

Research Issues in Data Mining and Knowledge Discovery, Dallas, Texas, USA, May 14 (2000).

[180] Stanoi, I., Riedewald, M., Agrawal, D., and Abbadi, A. E.: Discovery of Influence Sets in Frequently Updated Databases. In Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001), Roma, Italy, September 11-14 (2001).

[181] StatLib-Datasets Archive http://lib.stat.cmu.edu/datasets/

[182] http://www.stern.nyu.edu/~aweigend/Time-Series/Data/

[183] Stollnitz, E.J. DeRose, T.D. and Salesin, D.H.: Wavelets for Computer Graphics: A Primer, Part I. IEEE Computer Graphics and Applications (1995).

[184] Sun, S., and Zhou, X. : Semantic Caching for Web-Based Spatial Applications. in Proceeding of APWeb 2005 (LNCS 3399), pages 783-794, March 29- April 1, 2005, Shanghai, China (2005).

[185] Taniar, D.: Data Mining and Knowledge Discovery Technologies (Advances in Data Warehousing and Mining). IGI Publishing, (2008).

[186] Theodoridis, S., and Koutroumbas, K.: Pattern Recognition. Amsterdam, The Netherlands: Academic Press (2003).

[187] Toharia, P., Robles, O.D., Rodríguez, A., Pastor, L.: A study of Zernike Invariants for Content-based image Retrieval. In: Mery, D., Rueda, L. (eds.) PSIVT 2007. LNCS, vol. 4872. Springer, Heidelberg (2007).

[188] Trobec, R.,Vajtersic,M. and Zinterhof,P. : Parallel Computing-Numerics, Applications, and Trends, Springer (2009).

[189] Turk M., and Pentland: A. Eigenfaces for Recognition, Journal of Cognitive Neuroscience (1991).

[190] UCR Time Series datasets http://www.cs.ucr.edu/~eamonn/time_series_data/

[191] Uhlmann, J. K.: Satisfying General Proximity/Similarity Queries with Metric Trees. Information Processing Letters, 40(4). Elsevier (1991).

[192] Ukkonen, E.: Approximate String-matching with q-grams and Maximal Matches. Theoretical Computer Science 92 (1992).

[193] Velichko,V.M., and Zagoruyko,N.G.: Automatic Recognition of 200 Words. Int. J. Man-Mach. Stud., 2 (1970).

[194] Vidal, E.,: An Algorithm for Finding Nearest Neighbors in (Approximately) Constant Average Time. Pattern Recognition Letters, 4, (1986).

[195] Vlachos, M., Gunopulos, D., Das, G.: Indexing Time-Series under Conditions of Noise. Data Mining in Time Series Databases, World Scientific Publishing (2004).

[196] Vlachos, M., Kollios, G., and Gunopulos, D.: Discovering Similar Multidimensional Trajectories. In Proc. 18th Int. Conf. on Data Engineering (2002).

[197] Vlachos, M., Lin, J., Keogh, E., Gunopulos, D.: Multi-Resolution K-Means Clustering of Time Series and Applications to Images. Workshop on Multimedia Data Mining (MDM), SIGKDD, Washington DC, (2003).

[198] Vogiatzis, D., Tsapatsoulis, N.: Missing Value Estimation for DNA Microarrays with Mutliresolution Schemes. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Vol. 4132/2006, Book: Artificial Neural Networks - ICANN (2006).

[199] Wagner, R.,A., Fischer, M. J.: The String-to-String Correction Problem. Journal of the Association for Computing Machinery, Vol. 21, No. I, (1974).

[200] Wang, Q., Megalooikonomou, V., and Faloutsos, C.: Time Series Analysis with Multiple Resolutions. Inf. Syst. 35, 1 (2010).

[201] Weber, R., Böhm, K.: Trading Quality for Time with Nearest-neighbor Search; in: Proceedings of the 7th International Conference on Extending Database Technology (EDBT 2000), Konstanz, Germany, in: Lecture Notes in Computer Science, vol. 1777, Springer (2000).

[202] Weber, R., Schek, H.-J., Blott, S.: A Quantitative Analysis and Performance Study for Similarity-search Methods in High-dimensional Spaces. in: Proceedings of the 24th International Conference on Very Large Data Bases VLDB'98, New York City, NY (1998).

[203] Wei, L., Keogh, E., and Xi, X.: SAXually Explict Images: Finding Unusual Shapes. ICDM (2006).

[204] Weinberger, K., Saul, L.: Distance Metric Learning for Large Margin Nearest Neighbor Classification. The Journal of Machine Learning Research 10 (2009) .

[205] Weisstein, E.W.: Mathworld. A Wolfram web-resource – http://mathworld.wolfram.com/.

[206] White D.A., Jain R.: Similarity Indexing with the SS-tree. Proc. 12th Int. Conf on Data Engineering, New Orleans, LA (1996).

[207] Witten, I.H., Frank, E.: Data Mining Practical machine learning Tools and Techniques, Second Edition, Elsevier (2009).

[208] Wu, Y.-L., Agrawal, D., and Abbadi, A.E.: A Comparison of DFT and DWT Based Similarity Search in Time-series Databases. In Proc. 9th Int. Conf. on Information and Knowledge Management (2000).

[209] Yang, C., and Lin, K.-I.: An Index Structure for Efficient Reverse Nearest Neighbor Queries. In Proceedings of the 17th International Conference on Data Engineering (ICDE 2001), Heidelberg, Germany, April 2-6, 2001. IEEE Computer Society (2001).

[210] Yianilos P.: Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces', Proc. 4th ACM-SIAM Symposium on Discrete Algorithms, SODA`93 (1993).

[211] Yianilos, P. N.: Excluded Middle Vantage Point Forests for Nearest Neighbor Ssearch. In Proceedings of the 6th DIMACS Implementation Challenge: Near Neighbor Searches (ALENEX1999), Baltimore, Maryland, USA, January 15-16 (1999).

[212] Yi, B,K., and Faloutsos, C.: Fast Time Sequence Indexing for Arbitrary Lp norms. Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt (2000).

[213] Yi, B, K. Jagadish, H., and Faloutsos, C.: Efficient Retrieval of Similar Time Sequences under Time Warping. In ICDE 98 (1998).

[214] Zezula et al., :Similarity Search - The Metric Space Approach, Springer (2005).

[215] Zezula, P., Savino, P., Amato, G., Rabitti, F.: Approximate Similarity Retrieval with M-trees, The VLDB Journal 7 (4) (1998) .

[216] Zhong, S., and Ghosh, J.: HMMs and Coupled HMMs for Multi-channel EEG Classification. In IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'02), volume 2, IEEE Press, (2002).

[217] Zhou, X., Prasher, S., Sun, S., and Xu, K.: Multiresolution Spatial Databases: Making Web-based Spatial Applications Faster. In Proceedings of Asia-Pacific Web Conference (2004).

[218] Zloof, M. M.: Query by example. In AFIPS NCC (1975).

The Dissertation Publications

1- Muhammad Fuad, M.M, Marteau, P.F. : Enhancing the Symbolic Aggregate Approximation Method Using Updated Lookup Tables. 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems – KES 2010 , Lecture Notes in Computer Science, 2010, Volume 6276/2010, 8-10 September 2010, Cardiff, Wales, UK (2010)

2- Muhammad Fuad, M.M, Marteau , P.F. : Extending the Edit Distance Using Frequencies of Common Characters. 19th International Conference on Database and Expert Systems Applications - DEXA'08, Lecture Notes in Computer Science, 2008, Volume 5181/2008, 1-5 September,2008, Turin, Italy (2008).

3- Muhammad Fuad, M.M, Marteau , P.F. : Fast Retrieval of Time Series by Combining a Multiresolution Filter with a Representation Technique. The International Conference on Advanced Data Mining and Applications–ADMA2010, Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence, 2010, Volume 6440/2010, 19-21 November 2010, ChongQing, China (2010).

4- Muhammad Fuad, M.M, Marteau , P.F. : Multi-resolution Approach to Time Series Retrieval", Fourteenth International Database Engineering and Applications Symposium– IDEAS 2010 , 16-18 August, 2010, Montreal, QC, CANADA (2010).

5- 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 , 22-24 September 2010, Carnegie Mellon University, Pittsburgh, PA, USA (2010).

6- Muhammad Fuad, M.M, Marteau, P.F. : The Extended Edit Distance Metric. Sixth International Workshop on Content-Based Multimedia Indexing (CBMI 2008) 18-20th June, 2008, London, UK (2008).

7- Muhammad Fuad, M.M, Marteau, P.F. : The Multi-resolution Extended Edit Distance. Third International ICST Conference on Scalable Information Systems, Infoscale, 2008, ACM Digital Library, June 4-6, 2008. Vico Equense, Italy (2008).

8- Muhammad Fuad, M.M, Marteau, P.F. : The Prolonged Multi-Resolution Edit Distance. Dutch-Belgian Information Retrieval workshop, DIR-2008,April 14-15, 2008, Maastricht, the Netherlands (2008).

9- Muhammad Fuad, M.M, Marteau, P.F. : Towards a Faster Symbolic Aggregate Approximation Method", 5th International Conference on Software and Data Technologies – ICSOFT 2010, 22-24 July 2010, Athens, Greece (2010).

10- Muhammad Fuad, M.M, Marteau, P.F. : Une Distance d'Édition Etendue Multi Résolution. Atelier à EGC 2008, INRIA, Sophia Antipolis, France (2008).

11- Marteau, P.F., Muhammad Fuad, M.M , and Ménier , G. : Echantillonnage Adaptatif et Classification Supervisée de Séries Temporelles. EGC 2006, 6èmes Journées, Extraction et Gestion de Connaissances, Atelier 9 : Fouille de données temporelles. Jan 17 2006, Lille, France (2006).

Appendix

Let Σ be a finite alphabet, and let Σ^* be the set of strings on Σ. Let $f_i^{(S)}$ be the frequency of the character i in S, where . S is a string represented by Σ^*.

Let

$$D(S,T) = |S| + |T| - 2\sum_i min\left(f_i^{(S)}, f_i^{(T)}\right)$$

Then $\forall S_1, S_2, S_3$ we have;

$$D(S_1,S_2) \le D(S_1,S_3) + D(S_3,S_2) \qquad \text{(A-1)}$$

for all n, where n is the number of characters used to represent the strings.

Note: In the case when a character does not exist in one or more of the strings, we can assume that this (these) string(s) has (have) 0 frequency of the missing character.

Proof

We will prove the above lemma by induction.

i- $n = 1$

This is a trivial case. Given three strings; S_1, S_2, S_3 represented by the same character a. Let S_1^a, S_2^a, S_3^a be the frequency of a in S_1, S_2, S_3, respectively.

We have six configurations in this case:

1- $S_1^a \le S_2^a \le S_3^a$
2- $S_1^a \le S_3^a \le S_2^a$
3- $S_2^a \le S_1^a \le S_3^a$
4- $S_2^a \le S_3^a \le S_1^a$
5- $S_3^a \le S_1^a \le S_2^a$
6- $S_3^a \le S_2^a \le S_1^a$

We will prove that relation (A-1) holds in these six configurations.

1- $S_1^a \le S_2^a \le S_3^a$

In this case we have:

$$min\left(S_1^a, S_2^a\right) = S_1^a$$
$$min\left(S_1^a, S_3^a\right) = S_1^a$$
$$min\left(S_2^a, S_3^a\right) = S_2^a$$

$$D(S_1, S_2) \overset{?}{\le} D(S_1, S_3) + D(S_3, S_2)$$

By substituting the above values in this last relation we get:

$$S_1^a + S_2^a - 2S_1^a \overset{?}{\le} S_1^a + S_3^a - 2S_1^a + S_3^a + S_2^a - 2S_2^a$$
$$0 \overset{?}{\le} 2S_3^a - 2S_2^a$$

This is valid according to the stipulation of this configuration.

2- $S_1^a \le S_3^a \le S_2^a$

In this case we have:

$$min\left(S_1^a, S_2^a\right) = S_1^a$$
$$min\left(S_1^a, S_3^a\right) = S_1^a$$
$$min\left(S_2^a, S_3^a\right) = S_3^a$$

$$D(S_1, S_2) \overset{?}{\le} D(S_1, S_3) + D(S_3, S_2)$$

By substituting the above values in this last relation we get:

$$S_1^a + S_2^a - 2S_1^a \overset{?}{\le} S_1^a + S_3^a - 2S_1^a + S_3^a + S_2^a - 2S_3^a$$
$$0 \overset{?}{\le} 0$$

valid.

3- $S_2^a \le S_1^a \le S_3^a$

In this case we have:

$$min\left(S_1^a, S_2^a\right) = S_2^a$$
$$min\left(S_1^a, S_3^a\right) = S_1^a$$
$$min\left(S_2^a, S_3^a\right) = S_2^a$$
$$D(S_1, S_2) \overset{?}{\leq} D(S_1, S_3) + D(S_3, S_2)$$

By substituting the above values in this last relation we get:

$$S_1^a + S_2^a - 2S_2^a \overset{?}{\leq} S_1^a + S_3^a - 2S_1^a + S_3^a + S_2^a - 2S_2^a$$

$$0 \overset{?}{\leq} 2S_3^a - 2S_1^a$$

valid.

4- $S_2^a \leq S_3^a \leq S_1^a$

In this case we have:

$$min\left(S_1^a, S_2^a\right) = S_2^a$$
$$min\left(S_1^a, S_3^a\right) = S_3^a$$
$$min\left(S_2^a, S_3^a\right) = S_2^a$$
$$D(S_1, S_2) \overset{?}{\leq} D(S_1, S_3) + D(S_3, S_2)$$

By substituting the above values in this last relation we get:

$$S_1^a + S_2^a - 2S_2^a \overset{?}{\leq} S_1^a + S_3^a - 2S_3^a + S_3^a + S_2^a - 2S_2^a$$

$$0 \overset{?}{\leq} 0$$

valid.

5- $S_3^a \leq S_1^a \leq S_2^a$
In this case we have:

$$min\left(S_1^a, S_2^a\right) = S_1^a$$
$$min\left(S_1^a, S_3^a\right) = S_3^a$$

$$min\left(S_2^a, S_3^a\right) = S_3^a$$
$$D(S_1, S_2) \overset{?}{\leq} D(S_1, S_3) + D(S_3, S_2)$$

By substituting the above values in this last relation we get:

$$S_1^a + S_2^a - 2S_1^a \overset{?}{\leq} S_1^a + S_3^a - 2S_3^a + S_3^a + S_2^a - 2S_3^a$$

$$0 \overset{?}{\leq} 2S_1^a - 2S_3^a$$

valid.

6- $S_3^a \leq S_2^a \leq S_1^a$

In this case we have:

$$min\left(S_1^a, S_2^a\right) = S_2^a$$
$$min\left(S_1^a, S_3^a\right) = S_3^a$$
$$min\left(S_2^a, S_3^a\right) = S_3^a$$
$$D(S_1, S_2) \overset{?}{\leq} D(S_1, S_3) + D(S_3, S_2)$$

By substituting the above values in this last relation we get:

$$S_1^a + S_2^a - 2S_2^a \overset{?}{\leq} S_1^a + S_3^a - 2S_3^a + S_3^a + S_2^a - 2S_3^a$$

$$0 \overset{?}{\leq} 2S_2^a - 2S_3^a$$

valid

From 1-6 we conclude that the lemma is valid for $n = 1$.

ii- Let us assume that the lemma holds for $n-1$, where $n \geq 2$ and we will prove it for n.

Since the lemma holds for $n-1$ then:

$$D(S_1, S_2) \leq D(S_1, S_3) + D(S_3, S_2) \qquad \text{(A-2)}$$

where

$$D(S_1,S_2)=|S_1|+|S_2|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right)$$

$$D(S_1,S_3)=|S_1|+|S_3|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right)$$

$$D(S_3,S_2)=|S_3|+|S_2|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right)$$

When a new character is added the strings represented by $n-1$ characters become represented by n.

Let the frequency of the newly introduced character be $f_n^{(S_1)}, f_n^{(S_2)}, f_n^{(S_3)}$ in S_1,S_2,S_3 respectively.

We have six configurations of the newly added character:

7- $f_n^{(S_1)} \le f_n^{(S_2)} \le f_n^{(S_3)}$
8- $f_n^{(S_1)} \le f_n^{(S_3)} \le f_n^{(S_2)}$
9- $f_n^{(S_2)} \le f_n^{(S_1)} \le f_n^{(S_3)}$
10- $f_n^{(S_2)} \le f_n^{(S_3)} \le f_n^{(S_1)}$
11- $f_n^{(S_3)} \le f_n^{(S_1)} \le f_n^{(S_2)}$
12- $f_n^{(S_3)} \le f_n^{(S_2)} \le f_n^{(S_1)}$

We will prove that relation (A-1) holds in these six configurations.

7- $f_n^{(S_1)} \le f_n^{(S_2)} \le f_n^{(S_3)}$

In this case we have:

$$min\left(f_n^{(S_1)}, f_n^{(S_2)}\right) = f_n^{(S_1)}$$
$$min\left(f_n^{(S_1)}, f_n^{(S_3)}\right) = f_n^{(S_1)}$$
$$min\left(f_n^{(S_2)}, f_n^{(S_3)}\right) = f_n^{(S_2)}$$

$$D(S_1,S_2) \overset{?}{\le} D(S_1,S_3)+D(S_3,S_2)$$

$$|S_1|+|S_2|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right)+ f_n^{(S_1)} + f_n^{(S_2)} - 2\,min\left(f_n^{(S_1)}, f_n^{(S_2)}\right)\overset{?}{\leq}$$

$$|S_1|+|S_3|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right)+ f_n^{(S_1)} + f_n^{(S_3)} - 2\,min\left(f_n^{(S_1)}, f_n^{(S_3)}\right)$$

$$|S_3|+|S_2|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right)+ f_n^{(S_3)} + f_n^{(S_2)} - 2\,min\left(f_n^{(S_3)}, f_n^{(S_2)}\right)$$

$\Rightarrow$

$$|S_1|+|S_2|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right)+ f_n^{(S_1)} + f_n^{(S_2)} - 2\,f_n^{(S_1)}\overset{?}{\leq}$$

$$|S_1|+|S_3|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right)+ f_n^{(S_1)} + f_n^{(S_3)} - 2\,f_n^{(S_1)}$$

$$|S_3|+|S_2|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right)+ f_n^{(S_3)} + f_n^{(S_2)} - 2\,f_n^{(S_2)}$$

$\Rightarrow$

$$|S_1|+|S_2|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right)\overset{?}{\leq}$$

$$|S_1|+|S_3|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right)$$

$$|S_3|+|S_2|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right)+ 2\,f_n^{(S_3)} - 2\,f_n^{(S_2)}$$

Taking (A-2) into account , we get:

$$0\overset{?}{\leq} 2\,f_n^{(S_3)} - 2\,f_n^{(S_2)}$$

which is valid according to (7).

8- $f_n^{(S_1)} \leq f_n^{(S_3)} \leq f_n^{(S_2)}$

In this case we have

$$min\left(f_n^{(S_1)}, f_n^{(S_2)}\right)= f_n^{(S_1)}$$

$$min\left(f_n^{(S_1)}, f_n^{(S_3)}\right) = f_n^{(S_1)}$$
$$min\left(f_n^{(S_2)}, f_n^{(S_3)}\right) = f_n^{(S_3)}$$

$$D(S_1,S_2) \overset{?}{\leq} D(S_1,S_3) + D(S_3,S_2)$$

$$|S_1| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) + f_n^{(S_1)} + f_n^{(S_2)} - 2\,min\left(f_n^{(S_1)}, f_n^{(S_2)}\right) \overset{?}{\leq}$$
$$|S_1| + |S_3| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right) + f_n^{(S_1)} + f_n^{(S_3)} - 2\,min\left(f_n^{(S_1)}, f_n^{(S_3)}\right)$$
$$|S_3| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right) + f_n^{(S_3)} + f_n^{(S_2)} - 2\,min\left(f_n^{(S_3)}, f_n^{(S_2)}\right)$$
$$\Rightarrow$$

$$|S_1| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) + f_n^{(S_1)} + f_n^{(S_2)} - 2\,f_n^{(S_1)} \overset{?}{\leq}$$
$$|S_1| + |S_3| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right) + f_n^{(S_1)} + f_n^{(S_3)} - 2\,f_n^{(S_1)}$$
$$|S_3| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right) + f_n^{(S_3)} + f_n^{(S_2)} - 2\,f_n^{(S_3)}$$

$$\Rightarrow$$

$$|S_1| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) \overset{?}{\leq}$$
$$|S_1| + |S_3| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right)$$
$$|S_3| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right)$$

Which is valid according to (A-2).

9- $f_n^{(S_2)} \leq f_n^{(S_1)} \leq f_n^{(S_3)}$

In this case we have:

$$min\left(f_n^{(S_1)}, f_n^{(S_2)}\right) = f_n^{(S_2)}$$
$$min\left(f_n^{(S_1)}, f_n^{(S_3)}\right) = f_n^{(S_1)}$$
$$min\left(f_n^{(S_2)}, f_n^{(S_3)}\right) = f_n^{(S_2)}$$

$$D(S_1, S_2) \overset{?}{\leq} D(S_1, S_3) + D(S_3, S_2)$$

$$|S_1| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) + f_n^{(S_1)} + f_n^{(S_2)} - 2\,min\left(f_n^{(S_1)}, f_n^{(S_2)}\right) \overset{?}{\leq}$$
$$|S_1| + |S_3| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right) + f_n^{(S_1)} + f_n^{(S_3)} - 2\,min\left(f_n^{(S_1)}, f_n^{(S_3)}\right)$$
$$|S_3| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right) + f_n^{(S_3)} + f_n^{(S_2)} - 2\,min\left(f_n^{(S_3)}, f_n^{(S_2)}\right)$$

$$\Rightarrow$$

$$|S_1| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) + f_n^{(S_1)} + f_n^{(S_2)} - 2\,f_n^{(S_2)} \overset{?}{\leq}$$
$$|S_1| + |S_3| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right) + f_n^{(S_1)} + f_n^{(S_3)} - 2\,f_n^{(S_1)}$$
$$|S_3| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right) + f_n^{(S_3)} + f_n^{(S_2)} - 2\,f_n^{(S_2)}$$

$$\Rightarrow$$

$$|S_1| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) \overset{?}{\leq}$$
$$|S_1| + |S_3| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right)$$
$$|S_3| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right) + 2\,f_n^{(S_3)} - 2\,f_n^{(S_1)}$$

Taking (A-2) into account , we get:

$$0 \overset{?}{\leq} 2\,f_n^{(S_3)} - 2\,f_n^{(S_1)}$$

which is valid according to (9).

10- $f_n^{(S_2)} \le f_n^{(S_3)} \le f_n^{(S_1)}$

In this case we have:

$$min\left(f_n^{(S_1)}, f_n^{(S_2)}\right) = f_n^{(S_2)}$$
$$min\left(f_n^{(S_1)}, f_n^{(S_3)}\right) = f_n^{(S_3)}$$
$$min\left(f_n^{(S_2)}, f_n^{(S_3)}\right) = f_n^{(S_2)}$$

$$D(S_1,S_2) \overset{?}{\le} D(S_1,S_3) + D(S_3,S_2)$$

$$|S_1| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) + f_n^{(S_1)} + f_n^{(S_2)} - 2\,min\left(f_n^{(S_1)}, f_n^{(S_2)}\right) \overset{?}{\le}$$
$$|S_1| + |S_3| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right) + f_n^{(S_1)} + f_n^{(S_3)} - 2\,min\left(f_n^{(S_1)}, f_n^{(S_3)}\right)$$
$$|S_3| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right) + f_n^{(S_3)} + f_n^{(S_2)} - 2\,min\left(f_n^{(S_3)}, f_n^{(S_2)}\right)$$
$$\Rightarrow$$
$$|S_1| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) + f_n^{(S_1)} + f_n^{(S_2)} - 2\,f_n^{(S_2)} \overset{?}{\le}$$
$$|S_1| + |S_3| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right) + f_n^{(S_1)} + f_n^{(S_3)} - 2\,f_n^{(S_3)}$$
$$|S_3| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right) + f_n^{(S_3)} + f_n^{(S_2)} - 2\,f_n^{(S_2)}$$

$$\Rightarrow$$

$$|S_1| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) \overset{?}{\le}$$
$$|S_1| + |S_3| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right)$$
$$|S_3| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right)$$

Which is valid according to (A-2).

11- $f_n^{(S_3)} \le f_n^{(S_1)} \le f_n^{(S_2)}$

In this case we have.

$$min\left(f_n^{(S_1)}, f_n^{(S_2)}\right) = f_n^{(S_1)}$$
$$min\left(f_n^{(S_1)}, f_n^{(S_3)}\right) = f_n^{(S_3)}$$
$$min\left(f_n^{(S_2)}, f_n^{(S_3)}\right) = f_n^{(S_3)}$$

$$D(S_1, S_2) \overset{?}{\le} D(S_1, S_3) + D(S_3, S_2)$$

$$|S_1| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) + f_n^{(S_1)} + f_n^{(S_2)} - 2\,min\left(f_n^{(S_1)}, f_n^{(S_2)}\right) \overset{?}{\le}$$
$$|S_1| + |S_3| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right) + f_n^{(S_1)} + f_n^{(S_3)} - 2\,min\left(f_n^{(S_1)}, f_n^{(S_3)}\right)$$
$$|S_3| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right) + f_n^{(S_3)} + f_n^{(S_2)} - 2\,min\left(f_n^{(S_3)}, f_n^{(S_2)}\right)$$

$\Rightarrow$

$$|S_1| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) + f_n^{(S_1)} + f_n^{(S_2)} - 2 f_n^{(S_1)} \overset{?}{\le}$$
$$|S_1| + |S_3| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right) + f_n^{(S_1)} + f_n^{(S_3)} - 2 f_n^{(S_3)}$$
$$|S_3| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right) + f_n^{(S_3)} + f_n^{(S_2)} - 2 f_n^{(S_3)}$$

$\Rightarrow$

$$|S_1| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) \overset{?}{\le}$$
$$|S_1| + |S_3| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right)$$
$$|S_3| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right) + 2 f_n^{(S_1)} - 2 f_n^{(S_3)}$$

Taking (A-2) into account , we get:

$$0 \overset{?}{\leq} 2 f_n^{(S_1)} - 2 f_n^{(S_3)}$$

valid according to (11).

12- $f_n^{(S_3)} \leq f_n^{(S_2)} \leq f_n^{(S_1)}$

In this case we have:

$$\begin{aligned}
&min\left(f_n^{(S_1)}, f_n^{(S_2)}\right) = f_n^{(S_2)} \\
&min\left(f_n^{(S_1)}, f_n^{(S_3)}\right) = f_n^{(S_3)} \\
&min\left(f_n^{(S_2)}, f_n^{(S_3)}\right) = f_n^{(S_3)}
\end{aligned}$$

$$D(S_1, S_2) \overset{?}{\leq} D(S_1, S_3) + D(S_3, S_2)$$

$$\begin{aligned}
&|S_1| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) + f_n^{(S_1)} + f_n^{(S_2)} - 2\, min\left(f_n^{(S_1)}, f_n^{(S_2)}\right) \overset{?}{\leq} \\
&|S_1| + |S_3| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right) + f_n^{(S_1)} + f_n^{(S_3)} - 2\, min\left(f_n^{(S_1)}, f_n^{(S_3)}\right) \\
&|S_3| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right) + f_n^{(S_3)} + f_n^{(S_2)} - 2\, min\left(f_n^{(S_3)}, f_n^{(S_2)}\right)
\end{aligned}$$

$\Rightarrow$

$$\begin{aligned}
&|S_1| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) + f_n^{(S_1)} + f_n^{(S_2)} - 2 f_n^{(S_2)} \overset{?}{\leq} \\
&|S_1| + |S_3| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right) + f_n^{(S_1)} + f_n^{(S_3)} - 2 f_n^{(S_3)} \\
&|S_3| + |S_2| - 2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right) + f_n^{(S_3)} + f_n^{(S_2)} - 2 f_n^{(S_3)}
\end{aligned}$$

$\Rightarrow$

$$|S_1|+|S_2|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_2)}\right) \overset{?}{\leq}$$
$$|S_1|+|S_3|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_1)}, f_i^{(S_3)}\right)$$
$$|S_3|+|S_2|-2\sum_{i=1}^{n-1} min\left(f_i^{(S_3)}, f_i^{(S_2)}\right)+2f_n^{(S_2)}-2f_n^{(S_3)}$$

Taking (A-2) into account, we get:

$$0 \overset{?}{\leq} 2f_n^{(S_2)}-2f_n^{(S_3)}$$

which is true according to (12).

From 7-12 we conclude that the lemma is valid for n.

From i and ii, the lemma holds.