On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration

Eamonn Keogh Shruti Kasetty

Computer Science & Engineering Department
University of California - Riverside
Riverside, CA 92521

{eamonn,skasetty}@cs.ucr.edu

ABSTRACT

In the last decade there has been an explosion of interest in mining time series data. Literally hundreds of papers have introduced new algorithms to index, classify, cluster and segment time series. In this work we make the following claim. Much of this work has very little utility because the contribution made (speed in the case of indexing, accuracy in the case of classification and clustering, model accuracy in the case of segmentation) offer an amount of "improvement" that would have been completely dwarfed by the variance that would have been observed by testing on many real world datasets, or the variance that would have been observed by changing minor (unstated) implementation details.

To illustrate our point, we have undertaken the most exhaustive set of time series experiments ever attempted, re-implementing the contribution of more than two dozen papers, and testing them on 50 real world, highly diverse datasets. Our empirical results strongly support our assertion, and suggest the need for a set of time series benchmarks and more careful empirical evaluation in the data mining community.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications *Data Mining*.

Keywords

Time Series, Data Mining, Experimental Evaluation.

1. INTRODUCTION

In the last decade there has been an explosion of interest in mining time series data. Literally hundreds of papers have introduced new algorithms to index, classify, cluster and segment time series. In this work we make the following claim. Much of the work in the literature suffers from two types of experimental flaws, implementation bias and data bias (defined in detail below). Because of these flaws, much of the work has very little generalizability to real world problems.

In particular, we claim that many of the contributions made (speed in the case of indexing, accuracy in the case of classification and clustering, model accuracy in the case of segmentation) offer an amount of "improvement" that would have been completely dwarfed by the variance that would have been observed by testing on many real world datasets, or the variance that would have been observed by changing minor (unstated) implementation details.

In order to support our claim we have conducted the most exhaustive set of time series experiments ever attempted, re-implementing the contribution of more than 25 papers and testing them on 50 real word datasets. Our results strongly support our contention.

We are anxious that this work should not be taken as been critical of the data mining community. We note that several papers by the current first author are among the worst offenders in terms of weak experimental evaluation. While preparing the survey we read more than 340 data mining papers and we were struck by the originality and diversity of approaches that researchers have used to attack very difficult problems. Our goal is simply to demonstrate that empirical evaluations in the past have often been inadequate, and we hope this work will encourage more extensive experimental evaluations in the future.

For concreteness we begin by defining the various tasks that occupy the attention of most time series data mining research.

- **Indexing (Query by Content):** Given a query time series Q, and some similarity/dissimilarity measure $D(Q,C)$, find the nearest matching time series in database DB.
- **Clustering:** Find natural groupings of the time series in database DB under some similarity/dissimilarity measure $D(Q,C)$.
- **Classification:** Given an unlabeled time series Q, assign it to one of two or more predefined classes.
- **Segmentation:** Given a time series Q containing n datapoints, construct a model $\overline{Q}$, from K piecewise segments ($K << n$) such that $\overline{Q}$ closely approximates Q.

SIGKDD '02, Ju1y, 23-26 2002, Edmonton, Alberta, Canada.

Note that segmentation has two major uses. It may be performed in order to determine when the underlying model that created the time series has changed [19, 20], or segmentation may simply be performed to created a high level representation of the time series that supports indexing, clustering and classification [20, 30, 31, 37, 39, 42, 44, 46, 48, 52, 57].

As mentioned above, our experiments were conducted on 50 real world, highly diverse datasets. Space limitations prevent us from describing all 50 datasets in detail, so we simply note the following. The data represents the many areas in which time series data miners have investigated, including finance, medicine, biometrics, chemistry, astronomy, robotics, networking and industry. We also note that all data and code used in this paper is available for free by emailing the first author.

The rest of this paper is organized as follows. In Section 2 we survey the literature on time series data mining, and summarize some statistics about the empirical evaluations. In Section 3, we consider the indexing problem, and demonstrate with extensive experiments that many of the published results do not generalized to real world problems. Section 4 considers the problem of evaluating time series classification and clustering algorithms. In Section 5 we show that similar problems occur for evaluation of segmentation algorithms. Finally in Section 6 we summarize our findings and offer concrete suggestions to improve the quality of evaluation of time series data mining algorithms.

2. SURVEY

In order to assess the quality of empirical evaluation in the time series data mining community we begin by surveying the literature. Although we reviewed more than 340 papers, we only included the subset of 56 papers actually referenced in this work when assessing statistics about the number of datasets etc. The subset was chosen based on the following (somewhat subjective) criteria.

- Was the paper ever referenced? Self-citations were not counted. The rule was relaxed for paper published in the last year because of publishing delays. We used ResearchIndex (http://citeseer.nj.nec.com/cs) to make this determination.
- Was the paper published in a conference or journal likely to be read by a data miner? For example, several interesting time series data mining papers have appeared in medical and signal processing conferences, but are unlikely to come to the attention of the data mining community.

The survey is very comprehensive, but was not intended to be exhaustive. Such a goal would in any case be subjective (should a paper which introduces a new clustering algorithm, and mentions that it could be used for time series be included?). In general the papers come from high quality conferences and journals, including (SIG)KDD (11), ICDE (10), VLDB (5), SIGMOD/PODS (5), and CIKM (6).

Having obtained the 56 papers, we extracted various statistics (discussed below) from them about their empirical evaluation. In most cases this was easy, but occasionally a paper was a little ambiguous in explaining some feature of its empirical evaluation. In such cases we made an attempt to contact the author for clarification, and failing that, used our best judgment.

In presenting the results of the survey, we echo the caution of Prechelt, that "*while high numbers resulting from such counting cannot prove that the evaluation has high quality, low numbers (suggest) that the quality is low*" [47].

2.1 Size of Test Datasets

We recorded the size the test dataset for each paper. Where two or more datasets are used, we considered only the size of the largest.

The results are quite surprising; the median size of the test database was only 10,000 objects. Approximately 89% of the test databases could comfortably fit on a 1.44 Mb floppy disk.

2.2 Number of Rival Methods

Another surprising finding of the survey is the relative paucity of rival methods to which the contribution of the paper is compared. The median number is 1 (The average is 0.91), but this number includes very unrealistic strawman. For example many papers (including one by the current first author [31]) compare times for an indexing method to sequential scan where both are preformed *in main memory*. However, it is well understood sequential scan enjoys a tenfold speed up when performed on disk because any indexing technique must perform costly random access, whereas sequential scan can take advantage of an optimized linear traverse of the disk [32].

The limited number of rival methods is particularly troubling for papers that introduce a novel similarity measure. Although 29 of the papers surveyed introduce a novel similarity measure, only 12 of them compare the new measure to any strawman. The average number of rival similarity measures considered is only 0.97.

2.3 Number of Different Test Datasets

Although the small sizes of the test databases and the relatively scarcity of comparisons with rival methods is by itself troublesome, the most interesting finding concerns the number of datasets used in the experimental evaluation. On average, each contribution is tested on 1.85 datasets (1.26 real and 0.59 synthetic). This numbers are astonishingly low when you consider that new machine learning algorithms are typically evaluated on at least a dozen datasets [12, 33].

In fact, we feel that the numbers above are optimistic. Of the 30 papers that use two or more datasets, a very significant fraction (64%), use both stock market data and random walk data. However, we strongly believe these really should be counted as the same dataset. It is well known that random walk data can perfectly model stock market data is terms of all statistical properties, including variance, autocorrelation, stationarity etc [17, 53].

Work by the late Julian L. Simon suggested that humans find it impossible to differentiate between the two [53]. To confirm this finding we asked 12 professors at UCRs Anderson Graduate School of Management to look at Figure 1 and determine which three sequences are random walk, and which three are real S&P500 stocks. The confusion matrix is show in Table 1.

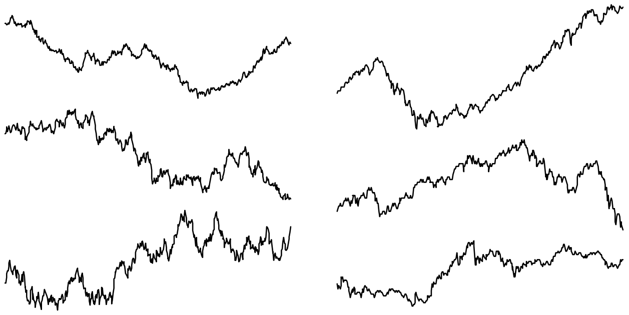

Figure 1. Six time series, three are random walk data, and three are real S&P500 stocks. Experiments suggest that humans cannot tell real and synthetic stock data apart (all the sequences on the right are real)

Table 1. The confusion matrix for human experts in attempting to differentiate between random walk data and stock market data

		Predicted	
		S&P Stock	**Random Walk**
Actual	**S&P Stock**	20	16
	Random Walk	16	20

The accuracy of the humans was 55.6%, which does not differ significantly from random guessing.

Given the above, if we consider stock market and random walk data to be the same, each paper in the survey is tested on average on only 1.28 different datasets. This number might be reasonable if the contribution had being claimed for only a single type of data [19, 37], or it had been shown that the choice of dataset has little influence on the outcome. However, the choice of dataset has a huge effect on the performance of time series algorithms. We will demonstrate this fact in the next 3 sections of this work.

3. INDEXING (QUERY BY CONTENT)

Similarity search in time series databases has emerged as an area of active interest since the classic first paper by Agrawal et al. [1]. More than 68% of the indexing approaches surveyed here use the original GEMINI framework of Faloutsos [17], but suggest a different approach to the dimensionality reduction stage. The proposed representations include the Discrete Fourier Transform (DFT) [1, 11, 16, 28, 49, 50], several kinds of Wavelets (DWT) [10, 27, 45, 51, 57, 60], Singular Value Decomposition [32, 35], Adaptive Piecewise Constant Approximation [32], Inner Products [18] and Piecewise Aggregate Approximation (PAA) [61]. The majority of work has focused solely on performance issues, however some authors have also considered other issues such as supporting non Euclidean distance measures [32, 50, 61] and allowing queries of arbitrary length [32, 40, 61].

3.1 Implementation Bias

Since most time series indexing techniques use the same indexing framework, and achieve the claimed speedup solely with the choice of representation, it is important to compare techniques in a manner that is free of implementation bias.

> **Definition**: *Implementation bias* is the conscious or unconscious disparity in the quality of implementation of a proposed approach, vs. the quality of implementation of the completing approaches.

Implementing fairly complex indexing techniques allows many opportunities for implementation bias. For example, suppose you hope to demonstrate that DWT is superior to DFT. With shift-normalized data [11, 28] the first DWT coefficient is zero so you could take advantage of that fact by indexing the 2nd to N+1th coefficients, rather than the 1st to Nth coefficients. However, you might neglect doing a similar optimization for DFT, whose first real coefficient is also zero for normalized data. Another possibility is that you might use the simple O(n^2) DFT algorithm rather than spend the time to code the more complex O(nLog$n$) radix 2 algorithm [32]. In both these cases DFT's performance would be artificially deflated relative to DWT.

One possible solution to the problem of implementation bias is extremely conscientious implementations of all approaches, combined with diligent explanations of the experimental process. Another possibility, which we explain below, is to design experiments that are free from the possibility of implementation bias.

Since all the exact indexing techniques use the same basic framework, the efficiency of indexing depends only on how well the dimensionality-reduced approximation can model the distances between the original objects. We can measure this by calculating the tightness of the lower bounds for any given representation.

> **Definition**: The *tightness of the lower bound* (denoted T) for any given representation is the ratio of the estimated distance between two sequences under that representation, over the true distance between the same two sequences.

Note that T is in the range [0,1]. A value of 1 would allow a constant time search algorithm, and a value of 0 would force the indexing structure to degrade to sequence scan. In fact, because sequential scan can take advantage of a linear traverse of the disk, whereas any indexing scheme must make wasteful random disk accesses, it is well understood that T must be significantly greater that 0 if we are to use the representation to beat sequential scan 32]. Since one can always create artificial data for any representation that will give an arbitrary value of T, it should be estimated *for a particular dataset* by random sampling. Note that the value of T for any given dimensionality reduction technique depends only on the data and is independent of any implementation choices such as page size, buffer size, computer language, hardware platform, seek time etc. A handful of papers in the survey already make use of a similar measure to compare the quality of representations [10, 32].

This idea of an implementation free evaluation of performance is by no means new. In artificial intelligence, researchers often compare search algorithms by reporting the number of nodes expanded, rather than the CPU times [33]. The problem of implementation bias is also well understood in other computer science domains, including parallel processing [5].

3.2 Data Bias

As mentioned above, the tightness of the lower bound can be estimated by random sampling of a dataset. However we have not yet addressed the importance of *which* dataset(s) are sampled. The indexing papers included in this survey tested their approach on a median of 1 datasets. This would be reasonable if the utility of the approach was only being claimed for a single type of data, for example "*More Efficient Indexing of ECG Time Series*" or "*A New Approach to Indexing Stock Market Data*". However, none of the papers make such a limited claim. The papers are implicitly or explicitly claiming to be improvements over the state of the art on *any* time series data. In fact, the choice of test data has a great effect on the experimental results, and virtually all papers surveyed suffer from data bias.

Definition: *Data bias* is the conscious or unconscious use of a particular set of testing data to confirm a desired finding.

There does not appear to be a simple cure for data bias. One possibility is to limit the scope of the claim for a new approach to that which has actually been demonstrated, e.g "*Faster indexing of Stock Market Data*". Another possibility, which we favor, is to test the algorithms on a large, heterogeneous set of time series. Ideally this set should include data that covers the spectrum of time series properties; stationarity/ non-stationarity, noisy/ smooth, cyclical/ non-cyclical, symmetric/ asymmetric, etc.

3.3 Empirical Demonstration of Implementation and Data Bias

To demonstrate the need for an implementation-free measure of the quality of indexing technique, and the absolute necessity of testing new algorithms on several datasets, consider the following contradictory claims made with regard the relative indexing abilities of DFT and DWT (wavelets):

- "*Several wavelets outperform the Haar wavelet (and DFT)*" [45].
- "*DFT-based and DWT-based techniques yield comparable results in similarity search*" [60].
- "*Haar wavelets perform slightly better that DFT*" [27].

Which, if any, of these statements are we to believe? Because of the problems of implementation bias and the limited number of test datasets we feel little credence can be given to any of the claims. To demonstrate this we have performed a comprehensive series of experiments that show that the variance due to implementation bias and testing on different data can far outweigh the improvements claimed in the literature.

We calculated the value of *T* for both DFT and DWT. To ensure that we obtained good estimates we averaged over 100,000 randomly chosen subsequences from each dataset. For fairness we used the same 100,000 subsequences for each approach. To ensure randomness in our sampling technique we used true random numbers that were created by a quantum mechanical process [55].

3.3.1 Demonstration of data bias

The three papers listed above experimented on a maximum of 3 datasets. If we use that number of datasets we can demonstrate essentially any finding we wish. For example, by working with the Powerplant, Infrasound and Attas datasets we can find that DFT outperforms the Haar wavelet, as shown in Figure 2.

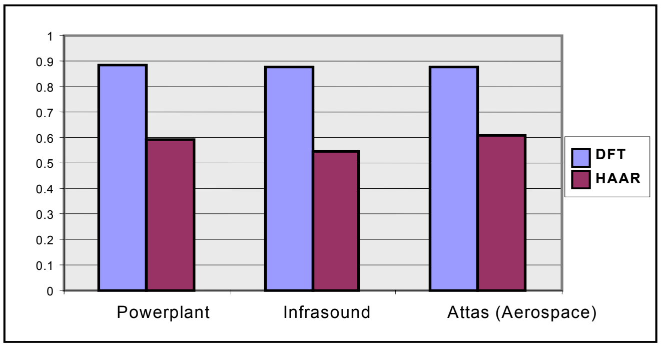

Figure 2. Experiments on the Powerplant, Infrasound and Attas datasets "demonstrate" that DFT outperforms DWT-Haar for indexing time series

In contrast if we worked with the Network, ERPdata and Fetal EEG datasets we could conclude that there is no real difference between DFT and Haar, as suggested by Figure 3.

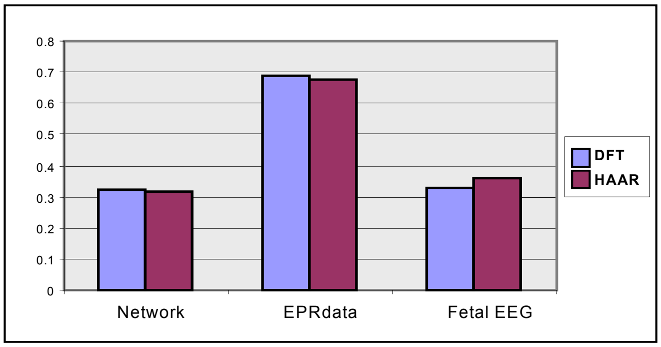

Figure 3. Experiments on the Network, EPRdata and Fetal EEG datasets "demonstrate" that DFT and DWT-Haar have the same performance for indexing time series

Finally had we had chosen the Chaotic, Earthquake and Wind datasets we could use the graphs in Figure 4 to demonstrate "convincingly" that Haar is superior to DFT.

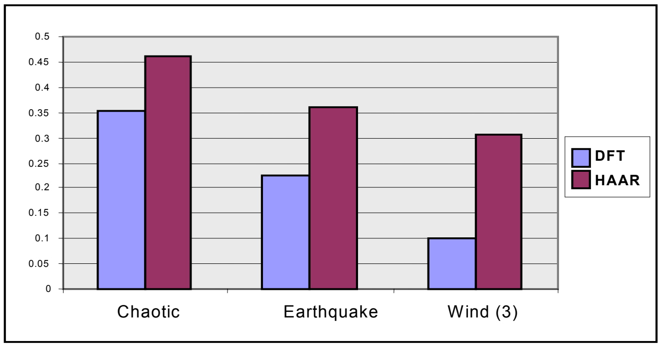

Figure 4. Experiments on the Chaotic, Earthquake and Wind datasets "demonstrate" that DWT-Haar outperforms DFT for indexing time series

Although we used the value of *T* to demonstrate the problem, we also confirmed the findings on an implemented system, using an

R-tree running on AMD Athlon 1.4 GHZ processor, with 512 MB of physical memory and 57.2 GB of secondary storage. The results were essentially identical, so we omit the graphs for brevity.

Note that we are not claiming any duplicity by the authors of the excellent papers listed above. We are merely demonstrating that the limited number of datasets used in the typical indexing paper severely limits the claims one can make.

3.3.2 Demonstration of implementation bias

The vast majority of papers on indexing that do use a strawman comparison use the simplest possible one, sequential scanning. Here we will demonstrate the potential for implementation bias with sequential scanning performed in main memory.

The Euclidean distance function is shown in Eq. 1.

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \qquad (1)$$

The basic sequential search algorithm is shown in Table 2.

Table 2. The Sequential Search Algorithm

```
Algorithm sequential_scan(data,query)
best_so_far = inf;
for every item in the database
  if euclidean_dist(data$_i$,query) < best_so_far
     pointer_to_best_match = i;
     best_so_far = euclidean_dist(data$_i$,query);
  end;
end;
```

One possibility implementation, which we call *Naïve*, is to calculate the Euclidean distance as shown in Eq. 1 with a loop to accumulate all the partial sums, followed by the taking of the square root. A possibility for optimization, which we call *Opt1*, is to neglect taking the square root. Since the square root function is monotonic, the ranking of the nearest neighbors will be identical under this scheme [61]. Finally we consider another optimization, which we call *Opt2*, which is simply to keep comparing the *best_so_far* variable to the partial sums at each iteration of the loop. If the partial sum ever exceeds the value of *best_so_far* we can admissibly abandon that calculation, since the partial distance can never decrease. To test the effect of these minor implementation details we performed 1-nearest neighbor searches in a random walk dataset, with a query length of 512 for increasingly larger datasets. The results are shown in Figure 5.

It is obvious that these very minor implementation details can produce large differences. If we are comparing a novel algorithm to sequential scan, and omit details of sequential scan implementation, it would be very hard to gauge the merit of our contribution. Note that for simplicity we only considered a main memory search. If we consider a disk-based search, there are a myriad of other implementation details that could effect the performance of sequential scan by at least an order of magnitude.

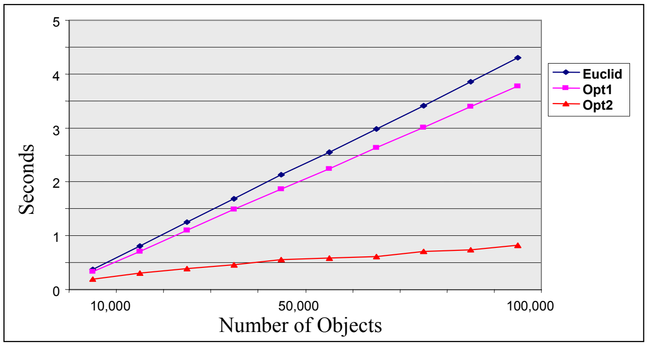

Figure 5. The affect of minor implementation details on the performance of sequential scan, for increasing large databases

It is easy to find examples of data bias in the literature, it is much more difficult to know the scale of the problem for implementation bias. By its very nature, it is almost impossible to know what fraction of a claimed improvement should be credited to the proposed approach, and what fraction may be due to implementation bias. However, there are a handful of examples where this is clear. For example, one paper included in the survey finds a significant performance gap between the indexing abilities of Haar wavelets and Piecewise Aggregate Approximation (PAA) [45]. However it was proved by two independent groups of researchers that these two approaches have exactly the same tightness of lower bounds when the number of dimensions is a power of two (and very little difference when the number of dimensions is not a power of two) [32, 61]. We empirically confirmed this fact 4,000,000 times during the experiments in Section 3.3.1. While there may be small differences in the CPU time to deal with the two representations, the order in which the original sequences are retrieved from disk by the index structure should be the same for both approaches, and disk time completely dominates CPU for time series indexing under Euclidean distance. We strongly suspect the spurious result reported above was the result of implementation bias, so we conducted an experiment to demonstrate how a simple implementation detail could produce an effect which is larger than the approximately 11% difference claimed.

We began our experiment by performing a fair comparison of the tightness of lower bounds for Haar and PAA on each of our 50 datasets, with a query length of 256 and 8 dimensions. Rather than estimate *T* with 100,000 random samples as in Section 3.1.1, we averaged over 100 samples as in the paper in question.

We repeated the experiment once more; this time neglecting to take advantage of the fact the first Haar coefficient is zero for normalized data. In other words, we wastefully index a value that is a constant zero. Once again we estimated *T* by averaging over 100 samples for each dataset.

For each dataset we calculated the ratio of the correct implementation's value of *T* to the poor implementation's value of *T*. The 50 results are plotted as a histogram in Figure 6.

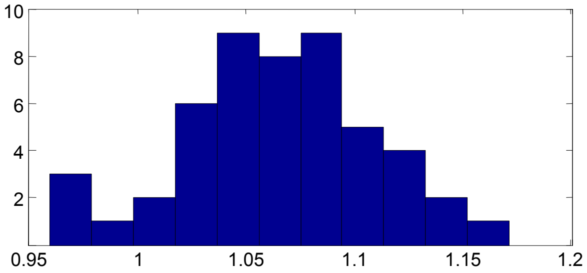

Figure 6. The distribution of the ratios of the results of correctly implemented experiments to experiments that have a slight implementation bias

It is surprising to note that sometimes implementation bias that should favor an approach can actually hurt it, as happened 4 times out of the 50 experiments. However we must remember that the values of *T* for each dataset were only estimated from 100 samples, and the finding is not statistically significant. What is clear from the experiment however is that a simple minor implementation detail can produce effects that are as large as the claimed improvement of the proposed approach

4. CLASSIFICATION AND CLUSTERING

Classification and clustering problems have been the subject of active research for decades [12, 33]. However the unique structure of time series means that most classic machine learning algorithms to not work well for time series. In particular the high dimensionality, very high feature correlation, and the (typically) large amounts noise that characterize time series data have been viewed as an interesting research challenge. Most of the contributions focus on providing a new similarity measure as a subroutine to an existing classification or clustering algorithm, so for simplicity we shall only consider the contribution of the suggested similarity measure.

How well do these similarity measures capture the true similarity of time series? There are two ways to answer this question, subjectively and objectively, we consider both below.

4.1 Subjective Evaluation of Similarity

Since a goal of data mining is often to find patterns that map onto human intuition, one possible way to judge the utility of a similarity measure is to show examples of time series that the proposed measure found to be similar/dissimilar. Surprisingly, many of the papers included in the survey, whose main contribution was to introduce a new similarity measure, fail to show even one example of a matching pair of time series [4, 8, 19, 22, 24, 26, 34, 36, 38, 42, 43, 48, 57]. Moreover, showing some examples of matching time series is of little utility unless some strawman comparison is used. Many papers ask us to consider the quality of their proposed similarity measure without a single comparison to another technique [2, 4, 8, 24, 31, 38, 39, 41, 42, 46, 57]. This in particularly surprising since the most obvious strawman, Euclidean distance, is trivial to implement (For example, in the Matlab programming language it requires only 19 characters: *sqrt(sum((q-c).^2))*).

We believe that one of the best (subjective) ways to evaluate a proposed similarity measure is to use it to create a dendrogram of several time series from the domain of interest [30].

Additional dendrograms can be created using other measures then plotted side by side with the propose approach. Figure 7 shows an example.

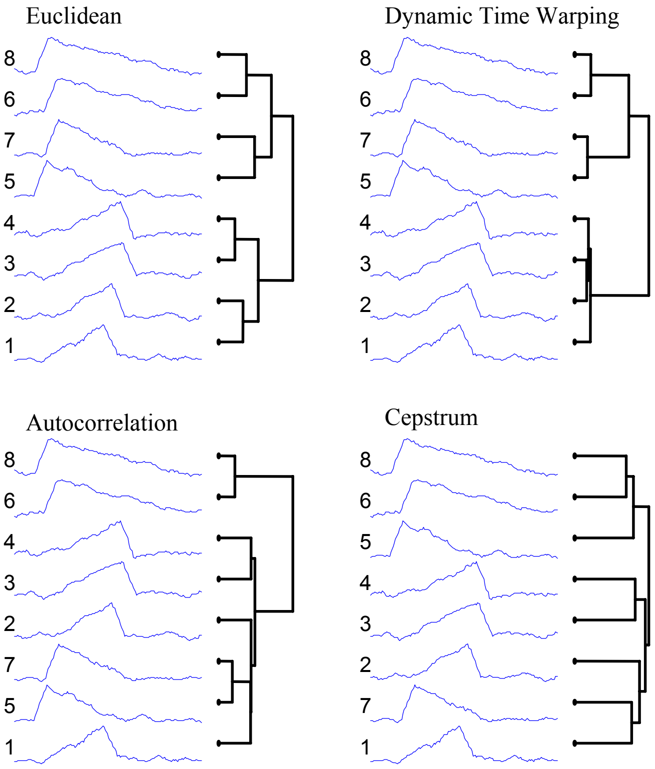

Figure 7. Dendograms can be used to visually assess the usefulness of a similarity measure. Above a dataset of 8 objects is clustered using the single linkage method, with 4 different distance measures. Euclidean distance and Dynamic Time Warping are decade old strawmen. The other two approaches have recently been proposed in data mining papers [57, 29]

Dendrograms are particularly attractive since a clustering of *M* objects summarizes O(*M*) measurements, however other possibilities of visualizing the quality of a similarity measure included projecting the time series into 2 dimensional space (via MDS or SOMs for example [15]).

4.2 Objective Evaluation of Similarity

Given a database of labeled time series, objective measurements of the quality of a proposed similarity measure can be readily obtained by running simple classification experiments. Although a few such databases do exist, very few advocates of a new similarity measure have chosen to demonstrate their contribution in this manner. The work by [21] is a notable exception. To repair this omission, we have undertaken an experimental comparison of many of the techniques included in the survey. We tested on two publicly available datasets:

- **Cylinder-Bell-Funnel**: This synthetic dataset has been in the literature for 8 years, and has been cited at least a dozen times [21]. It is a 3-class problem; we create 128 examples of each class for these experiments.
- **Control-Chart:** This synthetic dataset has been freely available for the UCI Data Archive since June 1998 [6]. It is a 6-class problem, with 100 examples of each class.

Note that for both problems, informal experiments suggest humans can achieve an error rate of zero. For simplicity we use the 1-Nearest Neighbor algorithm, evaluated using "leaving-one-out". We compare the proposed methods to the simplest strawman, Euclidean distance. This measure is well-known [1, 10, 11, 13, 14, 16, 17, 18, 27, 32, 35, 36, 40, 45, 49, 50, 60, 61, 62], parameterless, trivial to implement and predates data mining by several decades.

We originally intended to implement every proposed similarity measure in our survey, but several of the papers do not include a detailed enough description to allow reimplementation [39, 48]. We contented ourselves with reimplementing 11 measures. Some of the measures require the user to set some parameters. In these cases we wrapped the classification algorithm in a loop for each parameter, searched over all possible parameters and reported only the *best* result.

Table 3 summarized the results.

Table 3. The error rates for various similarity measures

Approach	Cylinder-Bell-Funnel	Control-Chart
Euclidean Distance	*0.003*	*0.013*
Aligned Subsequence [42]	0.451	0.623
Piecewise Normalization [26]	0.130	0.321
Autocorrelation Functions [57]	0.380	0.116
Cepstrum [29]	0.570	0.458
String (Suffix Tree) [24]	0.206	0.578
Important Points [46]	0.387	0.478
Edit Distance [8]	0.603	0.622
String Signature [4]	0.444	0.695
Cosine Wavelets [25]	0.130	0.371
Hölder [54]	0.331	0.593
Piecewise Probabilistic [31]	0.202	0.321

The results are quite surprising. None of the proposed techniques can beat the simple strawman. Their error rates are an order of magnitude worse that Euclidean distance. Several of the techniques have the error rates close to the default rate (i.e. the same error you would get randomly guessing). Although the inability to perform well on these two objective tests does not necessarily mean the similarity measures in question are without any merit (there may exist datasets on which they have reasonable accuracy), one has to wonder about the contribution of a new similarity measure which fails to demonstrate its utility on *any* objective or subjective test[1].

5. SEGMENTATION

A large fraction of the papers in the survey either introduce a segmentation algorithm as their main contribution, or utilize a segmentation algorithm as a subroutine. Although the segments created could be polynomials of an arbitrary degree, the most common representation of the segments are linear functions. Intuitively a Piecewise Linear Representation (PLR) refers to the approximation of a time series Q, of length n, with K straight lines. Figure 8 contains an example.

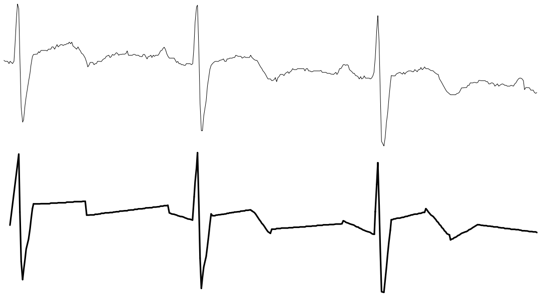

Figure 8. An example of a time series with its piecewise linear representation

Because K is typically much smaller that n, this representation makes the storage, transmission and computation of the data more efficient. Specifically, in the context of data mining, piecewise linear representation has been used to:

- Support novel distance measures for time series, including "fuzzy queries" [52], weighted queries [30], multiresolution queries [39, 48], dynamic time warping [42, 46], autocorrelation queries [57] and relevance feedback [30].
- Support concurrent mining of text and time series [37].
- Support novel clustering and classification algorithms [30].
- Support change point detection [20, 23].

Surprisingly, in spite of the ubiquity of this representation, with the exception of [52], there has been little attempt to understand and compare the algorithms that produce it.

Although appearing under different names and with slightly different implementation details, most time series segmentation algorithms can be grouped into one of the following three categories.

- **Sliding-Windows (SW)**: A segment is grown until it exceeds some error bound. The process repeats with the next data point not included in the newly approximated segment.
- **Top-Down (TD)**: The time series is recursively partitioned until some stopping criteria is met.
- **Bottom-Up (BU)**: Starting from the finest possible approximation, segments are merged until some stopping criteria is met.

We can measure the quality of a segmentation algorithm in several ways, the most obvious of which is to measure the reconstruction error for a fixed number of segments. The reconstruction error is simply the Euclidean distance between the original data and the segmented representation.

[1] Once again we wish to note that the current first author introduced one of the poorly performing measures.

5.1 Data Bias in Segmentation

Given that we have 3 algorithms to produce a segmented version of a time series, it is natural to ask which is best. The papers in the survey that use a segmentation algorithm test on a median of 1 dataset. However, if we use only one dataset we can demonstrate any finding we wish! There are 3 different algorithms, therefore 3! = 6 possible rankings. We tested the algorithms on our 50 fifty datasets, asking each algorithm to reduce a 1,024 datapoint time series to 64 segments. Amazingly, we found every possible ranking of the 3 algorithms as shown in Table 4.

Table 4. The 3 algorithms under consideration, ranked by reconstruction error (shown in brackets), on 6 datasets

Dataset	Best Algorithm	Second-Best Algorithm	Third-Best Algorithm
Soiltemp	TD (522.6)	SW (538.0)	BU (538.1)
Darwin	TD (575.2)	BU (821.0)	SW (833.9)
pHdata	SW (3.590)	TD (4.013)	BU (4.037)
Winding	SW (6.883)	BU (113.0)	TD (117.6)
Balloon	BU (168.1)	TD (224.5)	SW (234.1)
Network	BU (11.02)	SW (13.62)	TD (891.4)

Note that the fact that we could easily find datasets to demonstrate any ranking we wish does not preclude us from making a meaningful evaluation of the algorithms. In fact the Bottom-Up algorithm is significantly better than the other two approaches[2]. Our point, once again, is simply that little credence can be given to experimental results obtained from testing on a single dataset.

6. CONCLUSIONS AND RECOMMENDATIONS

In this work we have conducted a comprehensive survey of recent work on time series data mining. We have shown that because of several kinds of experimental flaws, in particular data bias and implementation bias, many of the results claimed in the literature have very little generalizability to real world problems. We have demonstrated our claim with the most comprehensive set of time series experiments ever undertaken.

Once again we would like to note that we view this work as a "call to arms" to the data mining community, and not a criticism of the many wonderful and original papers cited here. The intended spirit of this paper is similar to the ironically titled work by Bailey, "*Twelve ways to fool the masses when giving performance results on parallel computers*" [5]. The author later noted that few, if any researchers set out to deliberately mislead the academic community, but unless greater effort is made to meaningfully compare rival approaches, the entire field is in danger of being viewed with suspicion. This current work is an echo of that sentiment for the time series data mining community.

We conclude this paper with concrete suggestions for researchers working on time series data mining.

- Algorithms should be tested on a wide range of datasets, unless the utility of the approach is only been claimed for a particular type of data. If possible, one subset of the datasets should be used to fine tune the approach, then a different subset of the datasets should be used to do that the actual testing. This methodology is widely used in the machine learning community to help prevent implementation and data bias [12].
- Where possible, experiments should be designed to be free of the possibility of implementation bias. Note that this does not preclude the addition of extensive implementation testing.
- Novel similarity measures should be compared to simple strawmen, such as Euclidean distance or Dynamic Time Warping. Some subjective visualization, or objective experiments should justify their introduction.
- Where possible, all data and code used in the experiments should be made freely available to allow independent duplication of findings [6].

7. ACKNOWLEDGMENTS

The authors would like to thank Michael Pazzani, Pedro Domingos, Dimitrios Gunopulos and the anonymous reviewers for their valuable suggestions and comments. We also thank the many donors of test data.

8. REFERENCES

All papers except [5, 6, 12, 33, 47, 53], are included in the survey.

[1] Agrawal, R., Faloutsos, C. & Swami, A. (1993). Efficient similarity search in sequence databases. In *proceedings of the 4th Int'l Conference on Foundations of Data Organization and Algorithms*. Chicago, IL, Oct 13-15. pp 69-84.

[2] Agrawal, R., Lin, K. I., Sawhney, H. S. & Shim, K. (1995). Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *proceedings of the 21st Int'l Conference on Very Large Databases*. Zurich, Switzerland, Sept. pp 490-50.

[3] Agrawal, R., Psaila, G., Wimmers, E. L. & Zait, M. (1995). Querying shapes of histories. In *proceedings of the 21st Int'l Conference on Very Large Databases*. Zurich, Switzerland, Sept 11-15. pp 502-514.

[4] André-Jönsson, H. & Badal. D. (1997). 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. pp 211-220.

[5] Bailey, D. (1991). Twelve ways to fool the masses when giving performance results on parallel computers. *Supercomputing Review*, Aug. 1991, pp. 54-55.

[6] Bay, S. (1999). UCI Repository of Kdd databases [http://kdd.ics.uci.edu/]. Irvine, CA: University of California, Department of Information and Computer Science

[7] Berndt, D. J. & Clifford, J. (1996). Finding patterns in time series: a dynamic programming approach. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, Menlo Park, CA. pp 229-248.

[2] Bottom-Up outperformed Top-Down on 47 of 69 datasets, and it outperformed Sliding Windows on 58 of 69 datasets.

[8] Bozkaya, T., Yazdani, N. & Ozsoyoglu, Z. M. (1997). Matching and indexing sequences of different lengths. In *proceedings of the 6th Int'l Conference on Information and Knowledge Management*. Las Vegas, NV, Nov 10-14. pp 128-135.

[9] Caraça-Valente, J. P. & Lopez-Chavarrias, I. (2000). Discovering similar patterns in time series. *In proceedings of the 6th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data mining*. Boston, MA, Aug 20-23. pp 497-505.

[10] Chan, K. & Fu, A. W. (1999). Efficient time series matching by wavelets. In *proceedings of the 15th IEEE Int'l Conference on Data Engineering*. Sydney, Australia, Mar 23-26. pp 126-133.

[11] Chu, K. & Wong, M. (1999). Fast time-series searching with scaling and shifting. *In proceedings of the 18th ACM Symposium on Principles of Database Systems*. Philadelphia, PA, May 31-Jun 2. pp 237-248.

[12] Cohen, W. (1993). Efficient pruning methods for separate-and-conquer rule learning systems. In *proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France. pp 88-994.

[13] Das, G., Gunopulos, D. & Mannila, H. (1997). Finding similar time series. In *proceedings of Principles of Data Mining and Knowledge Discovery, 1st European Symposium*. Trondheim, Norway, Jun 24-27. pp 88-100.

[14] Das, G., Lin, K., Mannila, H., Renganathan, G. & Smyth, P. (1998). Rule discovery from time series. In *proceedings of the 4th Int'l Conference on Knowledge Discovery and Data Mining*. New York, NY, Aug 27-31. pp 16-22.

[15] Debregeas, A. & Hebrail, G. (1998). Interactive interpretation of kohonen maps applied to curves. In *proceedings of the 4th Int'l Conference of Knowledge Discovery and Data Mining*. New York, NY, Aug 27-31. pp 179-183.

[16] Faloutsos, C., Jagadish, H., Mendelzon, A. & Milo, T. (1997). A signature technique for similarity-based queries. In *proceedings of the Int'l Conference on Compression and Complexity of Sequences*. Positano-Salerno, Italy, Jun 11-13.

[17] Faloutsos, C., Ranganathan, M. & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. In *proceedings of the ACM SIGMOD Int'l Conference on Management of Data*. Minneapolis, MN, May 25-27. pp 419-429.

[18] Ferhatosmanoglu, H., Tuncel, E., Agrawal, D. & El Abbadi, A. (2001). Approximate nearest neighbor searching in multimedia databases. *In proceedings of the 17th IEEE Int'l Conference on Data Engineering*. Heidelberg, Germany, Apr 2-6. pp 503-511.

[19] Gavrilov, M., Anguelov, D., Indyk, P. & Motwani, R. (2000). Mining the stock market: which measure is best? In *proceedings of the 6th ACM Int'l Conference on Knowledge Discovery and Data Mining*. Boston, MA, Aug 20-23. pp 487-496.

[20] Ge, X. & Smyth, P. (2000). Deformable markov model templates for time-series pattern matching. In *proceedings of the 6th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*. Boston, MA, Aug 20-23. pp 81-90.

[21] Geurts, P. (2001). Pattern extraction for time series classification. In *proceedings of Principles of Data Mining and Knowledge Discovery, 5th European Conference*. Freiburg, Germany, Sept 3-5. pp 115-127.

[22] Goldin, D. & Kanellakis, P. (1995) On similarity queries for time-series data: constraint specification and implementation. In *proceedings of the 1st Int'l Conference on the Principles and Practice of Constraint Programming*. Cassis, France, Sept 19-22. pp 137-153.

[23] Guralnik, V. & Srivastava, J. (1999). Event detection from time series data. In *proceedings of the 5th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*. San Diego, CA, Aug 15-18. pp 33-42.

[24] Huang, Y. & Yu, P. S. (1999). 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. pp 282-286.

[25] Huhtala, Y., Kärkkäinen, J. & Toivonen, H. (1999). Mining for similarities in aligned time series using wavelets. *Data Mining and Knowledge Discovery: Theory, Tools, and Technology*, SPIE Proceedings Series, Vol. 3695. Orlando, FL, Apr. pp 150-160.

[26] Indyk, P., Koudas, N. & Muthukrishnan,S. (2000). Identifying representative trends in massive time series data sets using sketches. In *proceedings of the 26th Int'l Conference on Very Large Data Bases*. Cairo, Egypt, Sept 10-14. pp 363-372.

[27] Kahveci, T. & Singh, A. (2001). Variable length queries for time series data. In *proceedings of the 17th Int'l Conference on Data Engineering. Heidelberg*, Germany, Apr 2-6. pp 273-282.

[28] Kahveci, T., Singh, A. & Gurel, A. (2002). An efficient index structure for shift and scale invariant search of multi-attribute time sequences. In *proceedings of the 18th Int'l Conference on Data Engineering*. San Jose, CA, Feb 26-Mar 1. to appear.

[29] Kalpakis, K., Gada, D. & Puttagunta, V. (2001). Distance measures for effective clustering of ARIMA time-series. In *proceedings of the IEEE Int'l Conference on Data Mining*. San Jose, CA, Nov 29-Dec 2. pp 273-280.

[30] Keogh, E. & Pazzani, M. (1998). An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *proceedings of the 4th Int'l Conference on Knowledge Discovery and Data Mining*. New York, NY, Aug 27-31. pp 239-241.

[31] Keogh, E. & Smyth, P. (1997). A probabilistic approach to fast pattern matching in time series databases. In *proceedings of the 3rd Int'l Conference on Knowledge Discovery and Data Mining*. Newport Beach, CA, Aug 14-17. pp 24-20.

[32] Keogh, E., Chakrabarti, K., Pazzani, M. & Mehrotra, S. (2001). Locally adaptive dimensionality reduction for indexing large time series databases. In *proceedings of ACM SIGMOD Conference on Management of Data*. Santa Barbara, CA, May 21-24. pp 151-162.

[33] Kibler, D., & Langley, P. (1988). Machine learning as an experimental science. *In Proceedings of the 3rd European Working Session on Learning*. pp. 81-92

[34] Kim, E., Lam, J. M. & Han, J. (2000). AIM: approximate intelligent matching for time series data. In *proceedings of Data Warehousing and Knowledge Discovery, 2nd Int'l Conference*. London, UK, Sep 4-6. pp 347-357.

[35] Korn, F., Jagadish, H. & Faloutsos, C. (1997). Efficiently supporting ad hoc queries in large datasets of time sequences. *In*

proceedings of the ACM SIGMOD Int'l Conference on Management of Data. Tucson, AZ, May 13-15. pp 289-300.

[36] Lam, S. K. & Wong, M. H. (1998). A fast projection algorithm for sequence data searching. Data & Knowledge Engineering, Vol. 28(3). pp 321-339.

[37] Lavrenko, V., Schmill, M., Lawrie, D., Ogilvie, P., Jensen, D. & Allan, J. (2000). Mining of concurrent text and time series. In *proceedings of the 6th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining Workshop on Text Mining*. Boston, MA, Aug 20-23. pp 37-44.

[38] Lee, S., Chun, S., Kim, D., Lee, J. & Chung, C. (2000). Similarity search for multidimensional data sequences. In *proceedings of the 16th Int'l Conference on Data Engineering*. San Diego, CA, Feb 28-Mar 3. pp 599-608.

[39] Li, C., Yu, P. S. & Castelli, V. (1998). MALM: a framework for mining sequence database at multiple abstraction levels. In *proceedings of the 7th ACM CIKM Int'l Conference on Information and Knowledge Management*. Bethesda, MD, Nov 3-7. pp 267-272.

[40] Loh, W., Kim, S. & Whang, K. (2000). Index interpolation: an approach to subsequence matching supporting normalization transform in time-series databases. In *proceedings of the 9th ACM CIKM Int'l Conference on Information and Knowledge Management*. McLean, VA, Nov 6-11. pp 480-487.

[41] Park, S., Chu, W. W., Yoon, J. & Hsu, C. (2000). Efficient searches for similar subsequences of different lengths in sequence databases. In *proceedings of the 16th Int'l Conference on Data Engineering*. San Diego, CA, Feb 28-Mar 3. pp 23-32.

[42] Park, S., Kim, S. & Chu, W. W. (2001). Segment-based approach for subsequence searches in sequence databases. In *proceedings of the 16th ACM Symposium on Applied Computing*. Las Vegas, NV, Mar 11-14. pp 248-252.

[43] Park, S., Lee, D. & Chu, W. W. (1999). Fast retrieval of similar subsequences in long sequence databases. In *proceedings of the 3rd IEEE Knowledge and Data Engineering Exchange Workshop*. Chicago, IL, Nov 7.

[44] Polly, W. P. M. & Wong, M. H. (2001). Efficient and robust feature extraction and pattern matching of time series by a lattice structure. In *proceedings of the 10th ACM CIKM Int'l Conference on Information and Knowledge Management*. Atlanta, GA, Nov 5-10. pp 271-278.

[45] Popivanov, I. & Miller, R. J. (2002). Similarity search over time series data using wavelets. In *proceedings of the 18th Int'l Conference on Data Engineering*. San Jose, CA, Feb 26-Mar 1. pp 212-221.

[46] Pratt, K. B. & Fink, E. (2002). Search for patterns in compressed time series. Int'l Journal of Image and Graphics. to appear.

[47] Prechelt. L. (1995). A quantitative study of neural network learning algorithm evaluation practices. In *proceedings of the 4th Int'l Conference on Artificial Neural Networks*. pp. 223-227.

[48] Qu, Y., Wang, C. & Wang, X. S. (1998). Supporting fast search in time series for movement patterns in multiples scales. In *proceedings of the 7th ACM CIKM Int'l Conference on Information and Knowledge Management*. Bethesda, MD, Nov 3-7. pp 251-258.

[49] Rafiei, D. & Mendelzon, A. O. (1998). Efficient retrieval of similar time sequences using DFT. In *proceedings of the 5th Int'l Conference on Foundations of Data Organization and Algorithms*. Kobe, Japan, Nov 12-13.

[50] Refiei, D. (1999). On similarity-based queries for time series data. In *proceedings of the 15th IEEE Int'l Conference on Data Engineering*. Sydney, Australia, Mar 23-26. pp 410-417.

[51] Shahabi, C., Tian, X. & Zhao, W. (2000). TSA-tree: a wavelet based approach to improve the efficiency of multi-level surprise and trend queries. In *proceedings of the 12th Int'l Conference on Scientific and Statistical Database Management*. Berlin, Germany, Jul 26-28. pp 55-68.

[52] Shatkay, H. & Zdonik, S. (1996). Approximate queries and representations for large data sequences. In *proceedings of the 12th IEEE Int'l Conference on Data Engineering*. New Orleans, LA, Feb 26-Mar 1. pp 536-545.

[53] Simon, J. L. (1994). What some puzzling problems teach about the theory of simulation and the use of resampling. The American Statistician, Vol. 48(4). Nov. pp 1-4.

[54] Struzik, Z. & Siebes, A. (1999). The Haar wavelet transform in the time series similarity paradigm. In *proceedings of Principles of Data Mining and Knowledge Discovery, 3rd European Conference*. Prague, Czech Republic, Sept 15-18. pp 12-22.

[55] Walker, J. (2001). HotBits: Genuine random numbers generated by radioactive decay. www.fourmilab.ch/hotbits/

[56] Wang, C. & Wang, X. S. (2000). Multilevel filtering for high dimensional nearest neighbor search. In *proceedings of ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. Dallas, TX, May 14. pp 37-43.

[57] Wang, C. & Wang, X. S. (2000). Supporting content-based searches on time series via approximation. In *proceedings of the 12th Int'l Conference on Scientific and Statistical Database Management*. Berlin, Germany, Jul 26-28. pp 69-81.

[58] Wang, C. & Wang, X. S. (2000). Supporting subseries nearest neighbor search via approximation. In *proceedings of the 9th ACM CIKM Int'l Conference on Information and Knowledge Management*. McLean, VA, Nov 6-11. pp 314-321.

[59] Wu, L., Faloutsos, C., Sycara, K. & Payne, T. R. (2000). FALCON: feedback adaptive loop for content-based retrieval. In *proceedings of the 26th Int'l Conference on Very Large Data Bases*. Cairo, Egypt, Sept 10-14. pp 297-306.

[60] Wu, Y., Agrawal, D. & El Abbadi, A. (2000). A comparison of DFT and DWT based similarity search in time-series databases. In *proceedings of the 9th ACM CIKM Int'l Conference on Information and Knowledge Management*. McLean, VA, Nov 6-11. pp 488-495.

[61] Yi, B. & Faloutsos, C. (2000). Fast time sequence indexing for arbitrary lp norms. In *proceedings of the 26th Int'l Conference on Very Large Databases*. Cairo, Egypt, Sept 10-14. pp 385-394.

[62] Yi, B., Jagadish, H. & Faloutsos, C. (1998). Efficient retrieval of similar time sequences under time warping. In *proceedings of the 14th Int'l Conference on Data Engineering*. Orlando, FL, Feb 23-27. pp 201-20.