SFA: A Symbolic Fourier Approximation and Index for Similarity Search in High Dimensional Datasets

Patrick Schäfer
Zuse Institute Berlin
Berlin, Germany
patrick.schaefer@zib.de

Mikael Högqvist
Zuse Institute Berlin
Berlin, Germany
hoegqvist@zib.de

ABSTRACT

Time series analysis, as an application for high dimensional data mining, is a common task in biochemistry, meteorology, climate research, bio-medicine or marketing. Similarity search in data with increasing dimensionality results in an exponential growth of the search space, referred to as Curse of Dimensionality. A common approach to postpone this effect is to apply approximation to reduce the dimensionality of the original data prior to indexing. However, approximation involves loss of information, which also leads to an exponential growth of the search space. Therefore, indexing an approximation with a high dimensionality, i.e. high quality, is desirable.

We introduce Symbolic Fourier Approximation (SFA) and the SFA trie which allows for indexing of not only large datasets but also high dimensional approximations. This is done by exploiting the trade-off between the quality of the approximation and the degeneration of the index by using a variable number of dimensions to represent each approximation. Our experiments show that SFA combined with the SFA trie can scale up to a factor of 5–10 more indexed dimensions than previous approaches. Thus, it provides lower page accesses and CPU costs by a factor of 2–25 respectively 2–11 for exact similarity search using real world and synthetic data.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process; G.3 [**Probability and Statistics**]: Time series analysis; E.1 [**Data Structures**]: Trees

General Terms

Algorithms, Performance

Keywords

Time Series, Data Mining, Symbolic Representation, Discretisation, Indexing.

EDBT 2012, March 26–30, 2012, Berlin, Germany

1. INTRODUCTION

Time series databases, resulting from recording data over time, range from meteorological data like sediments from drill holes [20], financial data like stock prices or product sales, biomedical and biochemical data like ECG signals [14] or cellular networks [23]. Unlike exact search, similarity based search finds results that are similar to a query based on a similarity metric. Examples of similarity queries include:

- find all stocks that show *similar* trends
- find the patients with the 10 most *similar* ECGs
- find all products with a *similar* sales pattern.

A time series consisting of n measured values can be seen as a point in n-dimensional space, where the i-th measured value represents the i-th dimension. By indexing this n-dimensional point using a spatial index, the problem of finding *similar* time series is reduced to finding nearby points in n-dimensional space.

Indexing high dimensional data is a considerable challenge, as spatial index structures, like the R-Tree [3, 13] suffer from a phenomenon called the *Curse of Dimensionality*: with increasing dimensionality of the search space, the performance of similarity based queries on the index becomes worse than a linear scan of all data. Spatial index structures usually degenerate with 10–20 dimensions [12].

Work in [1, 11] introduces the idea of dimensionality reduction prior to indexing, and proved that, by using a *lower bounding distance measure*, queries are guaranteed to return the exact same result in reduced dimensional search space as if they were executed in the original space. In order to reduce the dimensionality of a time series an *approximation technique* is applied, effectively reducing dimensionality of the time series by 10:1 to 50:1. By use of an approximation technique the Curse of Dimensionality is shifted to 10–20 indexable dimensions of the approximations. The problem with approximation is that information from the original time series is lost. A goal of approximation is to find representations of the original data, so that each representation is distinct. For example, similar time series will have the same representation after an approximation if the reduced dimensionality is too low.

For two reasons the choice of the optimal dimensionality of the approximations is difficult:

1. Firstly, a higher dimensionality of the approximations helps to find distinct representations of the original time series. However, a spatial index degenerates exponentially with an increase of dimensions.

2. Secondly, with a high dimensionality we overrepresent dissimilar time series, while with a low dimensionality we underrepresent similar time series. Both of these issues impact the index negatively up to a point where a similarity search results in a sequential scan of the whole database.

We propose a technique which uses a variable number of dimensions for indexing time series approximations in order to postpone the impact of both issues. The idea is to group similar approximations based on a small common prefix. The length of the prefix is increased by 1 until each approximation in distinct. This can be implemented by using a trie, which is built over a set of strings. This introduces the problem of how to represent a time series as a string. Furthermore, it must be possible to extend the length of an approximation on the fly without the need to recalculate the approximation. This is why we introduce a symbolic representation based on the frequency domain as opposed to the spatial domain. In the frequency domain each dimension contains approximate information about the whole time series. By increasing the dimensionality we can add detail, thus improving the overall quality of the approximation. In the spatial domain we have to decide on a length of the approximation in advance and a prefix of this length only represents a subset of the time series.

In this paper we introduce a novel symbolic representation called *Symbolic Fourier Approximation* (SFA) and the *SFA trie*, an index structure utilising the properties of the frequency domain nature of SFA. As part of this technique we

- propose a symbolic representation based on *Discrete Fourier Transform* (DFT) for approximation and show the benefits compared to other techniques such as *Piecewise Aggregate Approximation* (PAA),
- introduce a novel discretisation technique called *multiple coefficient binning* (MCB) which improves pruning of the search space during the query execution,
- provide a proof of an *Euclidean lower bounding distance measure* for SFA which guarantees that the query results of a similarity search in SFA reduced space are the same as in the original space,
- introduce the *SFA trie*, a modification of a prefix tree built from the strings of the SFA approximations, and
- show through experiments that SFA and the SFA trie scale to a factor of 5–10 higher indexed dimensions than previous approaches and can index very large datasets. Furthermore, the SFA trie is better than previous approaches by up to a factor of 2–25 in terms of exact search performance on real and synthetic time series datasets.

The rest of the paper is organised as follows: Section 2 begins with related work and background material. Section 3 introduces our novel discretisation technique, our symbolic representation and the lower bounding distance measure. Section 4 presents the SFA trie. In Section 5 we perform exhaustive experiments on pruning power and indexing performance. In Section 6 we give a conclusion and suggest future work.

2. BACKGROUND AND RELATED WORK

2.1 Similarity Search in High Dimensions

A time series $C = c_1 \dots c_n$ of length n can be represented by a point in n-dimensional space. Finding similar time series is thus reduced to finding similar or nearby points in a high dimensional space.

The *similarity* of two points Q and C is expressed in terms of a real value using a distance measure $D_{true}(Q, C) \rightarrow \mathbb{R}$. The distance measure is application dependent. Similarity queries include nearest neighbour or epsilon-range queries, for example.

DEFINITION 1. *k-nearest neighbour (k-NN) query: a k-NN query for Q is a similarity query in n-dimensional space DS^n, which returns a subset $NN_k(Q) \subseteq DS^n$ that contains k objects with the following condition: $\forall t \epsilon NN_k(Q), \forall o \epsilon DS^n - NN_k(Q) : D_{true}(t, Q) \leq D_{true}(o, Q)$.*

Work in [1, 11] introduced a general framework called *GEMINI* for indexing high dimensional data: A time series $C = c_1 \dots c_n$ is mapped into a much lower dimensional space by the use of an approximation technique, called *dimensionality reduction*. This much lower dimensional space is indexed using a spatial index and a similarity query can be answered by traversing the index.

Approximation involves loss of information and the distance of any two points in original space is not preserved in lower dimensional space. To guarantee the absence of false dismissals, we have to define a *lower bounding distance measure* $D_{IndexSpace}(Q, C)$ in lower dimensional space, which underestimates the true distance $D_{true}(Q, C)$ between the time series Q and C in original space. This property is called the *lower bounding lemma* [11]:

$$D_{IndexSpace}(Q, C) \leqslant D_{true}(Q, C) \quad (1)$$

Similarity search can be divided into two categories:

1. *whole matching*: given N time series and a query for Q, all of length n, find the time series most similar to Q according to distance measure D_{true}.
2. *subsequence matching*: given a short query for Q of length m and a long time series, find all subsequences within the long time series similar to Q.

Subsequence matching can be reduced to whole matching by using a sliding window, which is shifted along the long time series to extract time series of length m which are then compared to Q.

2.2 Dimensionality Reductions

Dimensionality reductions can be divided into two groups: *symbolic* and *numerical*. In these methods a time series is represented as a sequence of discrete values (symbols) or real values respectively. Since symbolic representations are essentially a character string, they can also be used in data structures and algorithms in the field of data-mining such as tries, hashing, Markov models, string-matching [19]. Furthermore, they allow for indexing large datasets [5, 26].

The process of transforming a time series into a symbolic representation can be generalised to two parts:

1. *approximation* is applied to map a time series into lower dimensional space resulting in a vector of real values, and

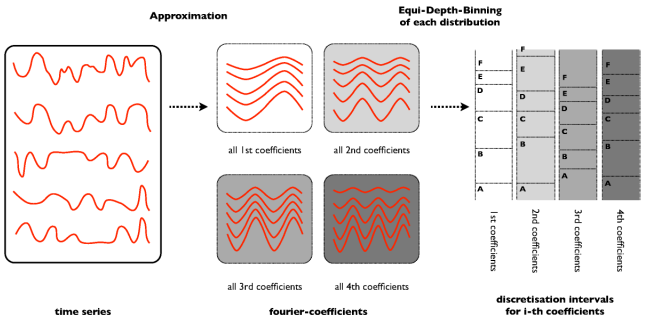


Figure 1: MCB applies binning to the distribution of all i-th coefficients.

2. *discretisation* is used to map each real value to a discrete value, which can be interpreted as a symbol.

Numerical dimensionality reductions, for which a lower bounding distance measure has been introduced, include Discrete Fourier Transform (DFT) [1, 11], Discrete Wavelet Transform (DWT) [8, 22, 7], Piecewise Aggregate Approximation (PAA) [16], Adaptive Piecewise Constant Approximation (APCA) [6], Chebyshev Polynomials (CHEBY) [4], Piecewise Linear Approximation (PLA) [9] or Singular Value Decomposition (SVD) [17]. With the exception of SVD, due to the runtime, we use all of these dimensionality reductions for our experimental comparison.

Symbolic Aggregate approXimation (SAX) [18] was the first symbolic representation to introduce a lower bounding distance measure. A time series is represented by a sequence of symbols using an alphabet of fixed size for each symbol. *Indexable SAX* (iSAX) [26, 5] is a modification of SAX, which allows for efficient indexing through the iSAX index. iSAX introduces a dynamic alphabet size. For each new level in the index, the alphabet size is doubled. The iSAX index and the SFA trie are complementary approaches, as the SFA trie uses an alphabet of fixed size for the fanout of the index and dynamically increases the quality of the approximation for splitting, while the iSAX index follows the opposite approach: fixed approximation length and dynamic alphabet size.

Space partitioning index structures like the R-tree [13] or TS-tree [2] apply partitioning to k-dimensional space to organise points. The search space is fixed to a constant dimensionality, thus all approximations are transformed using this dimensionality, and the space is split at a finer granularity at each new level of the index. This is opposed to the SFA trie where each approximation is computed at a high dimensionality, but a dynamic prefix of each approximation is used for splitting at each new level of the index.

3. SFA: SYMBOLIC REPRESENTATION

The symbolic representation SFA consists of two phases: A preprocessing phase which is performed using all time series and the transformation phase which is performed for each single time series using the discretisations obtained from preprocessing:

1. Preprocessing: determine MCB discretisation

   - DFT Approximation: all time series are approximated using DFT (Figure 1 left).

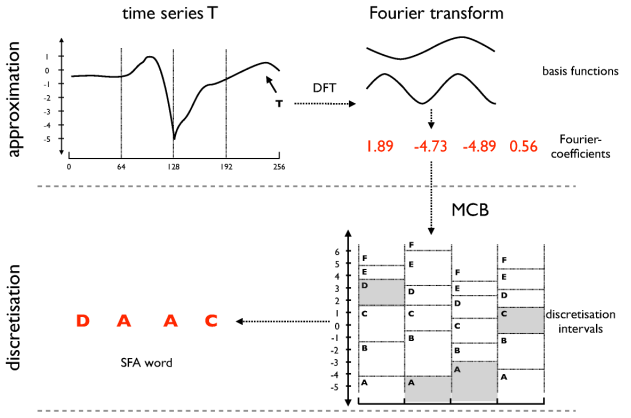


Figure 2: SFA: A time series is approximated and discretised using MCB discretisation.

   - MCB Discretisation: multiple discretisations are determined from all DFT approximations using a discretisation technique we call *multiple coefficient binning* (MCB) (Figure 1 right).

2. Transformation: apply MCB discretisation

   - SFA Transformation: each DFT approximation is discretised using those discretisations obtained from preprocessing. This results in the *SFA word* of a time series (Figure 2).

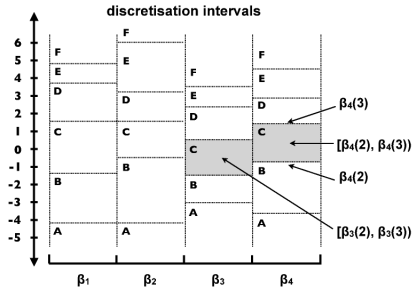
The basic rationale for using DFT for approximation is simple: a tighter lower bound of the approximation results in a tighter lower bound of the symbolic representation. This can be correlated to a reduction of false alarms when performing similarity search in lower dimensional space. DFT has a significantly tighter lower bound than PAA and therefore SFA has a tighter lower bound than SAX/iSAX (see Experiment 5.1.2). However, MCB as a discretisation technique is not limited to a single approximation technique like DFT. In this paper we use DFT since it shows the best results in our experiments in Section 5.1.3

A result of the usage of DFT, which introduces frequency domain, is that each frequency has its own distribution. This leads to the necessity to apply individual discretisation to each group of frequencies (see Experiment 5.1.3), which is the goal of MCB.

3.1 Preprocessing: DFT Approximation

The idea of *DFT* is to decompose a time series into a sum of (orthogonal) basis functions. The first few basis functions correspond to slowly changing sections and represent the coarse distribution, while later basis functions represent rapid changes like gaps or noise. Thus the use of only the first few basis functions produces a good approximation of a time series.

For *DFT* a time series T of length n is decomposed using sinusoid waves as the basis function. Each wave is represented by a complex number X_f called the *Fourier coefficient*, where the magnitude represents the amplitude and the phase represents the initial phase angle of the wave. A time series is approximated using the sequence of the first w coefficients



breakpoints

a	β_1(a)	β_2(a)	β_3(a)	β_4(a)
0	-∞	-∞	-∞	-∞
1	-4,09	-4,12	-3,31	-3,62
2	-1,29	-0,49	-1,5	-0,8
3	1,55	1,54	0,57	1,36
4	3,68	3,15	2,28	2,65
5	4,73	5,98	3,43	4,24
6	+∞	+∞	+∞	+∞

Figure 3: SFA discretisation intervals and corresponding breakpoints for the Koski ECG dataset, an alphabet of size $c = 6$ (symbols A-F) and a symbolic word length $w = 4$.

(each represented by real and imaginary part), with $w \ll n$:

$$DFT(T, w) \quad = \quad X_0 \dots X_{w-1}$$

See [1, 24] for details.

3.2 Preprocessing: MCB Discretisation

The aim of *multiple coefficient binning* (MCB) is to minimise the loss of information introduced by discretisation, since a better description of the original signals improves pruning during query execution.

This is done by applying w discretisations, given an SFA word length w, where the i-th discretisation is applied to the i-th coefficients of the DFT approximations. To obtain those discretisations, we arrange the i-th coefficients of all DFT approximations into the i-th group, with a total of w groups. A *histogram* is build for each group. Finally, binning is applied to each group (Figure 1), which gives the w distinct discretisations.

A *histogram* represents the distribution of values for each group of coefficients.

Definition 2. Histogram*: Given the approximations of N time series denoted by* DS*, with length of an approximation w, the value $H_i(x)$ represents the total number of approximations, for which the i-th coefficient $t_i \epsilon T$ is equal to x:*

$$H_i(x) = |\{T \mid t_i = x, T = (t_0 \dots t_{w-1}) \epsilon DS\}|, i \epsilon \{1, \dots, w\}$$

To obtain the MCB discretisation, equi-depth or equi-width binning is applied to each histogram.

Definition 3. MCB*: Given approximations of N time series and an alphabet $\Sigma = \{symbol_1, \dots, symbol_c\}$, MCB determines breakpoints $\beta_i(0) \leq \dots \leq \beta_i(c)$ such that we have for each histogram H_i:*

- *for* equi-depth binning*: the total number of time series falling into any interval $[\beta_i(a-1), \beta_i(a))$ is the same (i.e. $\sum_{\beta_i(a-1) \leq x < \beta_i(a)} H_i(x) = \frac{N}{c}$),*
- *for* equi-width binning*: the width of any interval $[\beta_i(a-1), \beta_i(a))$ is equal*

with the definitions $\beta_i(0) = \min\{t_i | T = (t_0 \dots t_{w-1}) \epsilon DS\}$ and $\beta_i(c) = \max\{t_i | T = (t_0 \dots t_{w-1}) \epsilon DS\}$.

Each of the w histograms is used to obtain $c+1$ breakpoints, where two breakpoints define a *discretisation interval.*

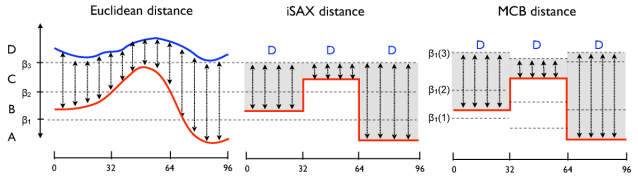


Figure 4: Euclidean distance of two time series C and Q compared to the Euclidean lower bounding distance using *iSAX* and *SFA (MCB).*

Definition 4. Discretisation*: Given an alphabet $\Sigma = \{symbol_1, \dots, symbol_c\}$ and breakpoints $\beta_i(0) \leq \dots \leq \beta_i(c)$ the a-th* discretisation interval *in the i-th group is defined by $DI_i(a) = [\beta_i(a-1), \beta_i(a))$ and labelled by $symbol_a$, derived from dividing each histogram H_i into $c+1$ breakpoints.*

Finally, the i-th discretisation *consists of the c discretisation intervals $DI_i(1), \dots DI_i(c)$.*

For an alphabet of size c, binning results in a total of $c \cdot w$ discretisation intervals, i.e. c symbols for each of the w histograms.

Figure 3 shows one possible set of 4 discretisations with 6 discretisation intervals each and the corresponding breakpoints for the Koski ECG dataset [14] when using an alphabet of size 6, a SFA word length of 4 and equi-depth binning.

3.3 SFA Transformation

Given the discretisations from preprocessing, the SFA word is obtained from each approximation by a simple lookup.

Definition 5. SFA Word*: The symbolic representation $SFA(T) = s_1 \dots s_w$ for a time series T with approximation $DFT(T, w) = \bar{t_1} \dots \bar{t_w}$, is a mapping $SFA : DS \rightarrow \Sigma^w$ for each time series to a sequence of symbols of the alphabet Σ. Specifically, the i-th numeric value $\bar{t_i}$ is mapped to the a-th symbol of the alphabet $\Sigma = \{symbol_1, \dots, symbol_c\}$, if it falls into the corresponding discretisation interval $DI_i(a)$:*

$$s_i \equiv symbol_a, \text{ iff } t_i \epsilon DI_i(a)$$

SAX/iSAX discretisation can be seen as a special case of MCB discretisation, where the breakpoints for all coefficients are equal $\beta_1 = \dots = \beta_w$ and equi-depth binning of Gaussian distribution is applied to determine the initial breakpoints β_1.

3.4 Lower Bounding Distance Measures

Within the context of time series analysis the true distance of two time series Q and C of length n is usually measured in terms of the Euclidean distance (Figure 4 left):

$$D^2_{true}(Q, C) = \sum_i (q_i - c_i)^2$$

Given the DFT representations of two time series $C_{DFT} = (c'_1 \dots c'_w)$ of C and $Q_{DFT} = (q'_1 \dots q'_w)$ of Q the DFT Euclidean lower bounding distance is defined as [24] (for $w \ll n$):

$$D^2_{DFT}(C_{DFT}, Q_{DFT}) = 2 \sum_{i=2}^{w} (c'_i - q'_i)^2 \qquad (2)$$

Note that the first DFT coefficients c'_1 and q'_1 have been discarded, as these are 0 for z-normalised time series.

The *SFA Euclidean lower bounding distance* between a DFT representation $Q_{DFT} = (q'_1 \dots q'_w)$ and an SFA representation $C_{SFA} = (c''_1 \dots c''_w)$ is calculated by exchanging the pairwise difference of the numerical values in Eq. 2 by a $dist_i$ function, which measures the distance between the i-th symbol and the i-th numerical value:

$$D^2_{SFA}(C_{SFA}, Q_{DFT}) \equiv 2\sum_{i=2} dist_i(c''_i, q'_i)^2 \quad (3)$$

The distance $dist_i$ between a numerical value q'_i and a symbol $c''_i = symbol_a$, represented by its lower and upper discretisation breakpoints $\beta_i(a-1)$ and $\beta_i(a)$, is defined as the distance to the lower discretisation breakpoint if q'_i is smaller or the upper discretisation breakpoint if q'_i is larger:

$$dist_i(c''_i, q'_i) \equiv \begin{cases} 0, & \text{if } q'_i \epsilon DI_i(a) \\ \beta_i(a-1) - q'_i, & \text{if } q'_i < \beta_i(a-1) \\ q'_i - \beta_i(a), & \text{if } q'_i > \beta_i(a) \end{cases} \quad (4)$$

Figure 4 (right) illustrates the difference between the iSAX *dist* and the MCB $dist_i$ definition. MCB uses different breakpoints while iSAX uses the same breakpoints for each symbol.

3.5 Proof of Correctness

To prove correctness of SFA approximation using Eq. 3 and Eq. 4, we have to show that the Euclidean lower bounding lemma holds for SFA.

Claim 1. The distance measure D^2_{SFA} for two time series $Q_{DFT} = q'_1 \dots q'_w$ and $C_{DFT} = c'_1 \dots c'_w$, $C_{SFA} = c''_1 \dots c''_w$ holds the Euclidean lower bounding lemma:

$$D^2_{SFA}(C_{SFA}, Q_{DFT}) \quad \leqslant \quad D^2_{True}(C, Q)$$

Proof. iSAX allows Euclidean lower bounding using distance $dist(c, q)$. It is obvious, following proof [19], that the distance $dist_i(c''_i, q'_i)$ using MCB breakpoints is still always smaller than the distance of two DFT-transformed coefficients q'_i and c'_i:

$$dist^2_i(c''_i, q'_i) \leq (c'_i - q'_i)^2$$

This yields

$$\begin{aligned} D^2_{true}(C, Q) &\geqslant D^2_{DFT}(C_{DFT}, Q_{DFT}) && (5) \\ &= 2\sum_i (c'_i - q'_i)^2 && (6) \\ &\geqslant 2\sum_i dist_i(c''_i, q'_i)^2 && (7) \\ &= D^2_{SFA}(C_{SFA}, Q_{DFT}) && (8) \end{aligned}$$

where Eq. 5 is proven and Eq. 6 is defined in [24]. □

3.6 Runtime Complexity

Claim 2. *The transformation of N SFA words of length w over an alphabet of size c from a set of N time series of length n has a complexity of $O(N \cdot n \log n)$.*

Proof. The calculation of the N DFT approximations requires $O(N \cdot n \log n)$ operations, resulting in N approximations of length w. The calculation of the MCB discretisation intervals requires $O(Nw)$ operations, as it involves scanning all approximations of length w. The N SFA words require Nw lookups in the breakpoints for the c intervals. Assuming we use binary search for those lookups this results in $O(Nw \cdot \log c)$ operations. This leads to a total of $O(N \cdot (n \log n + w + w \log c)) = O(N \cdot n \log n)$ operations for N time series (Table 1). □

This means the transformation is dominated by the N DFTs and there is negligible impact by the SFA word length w or alphabet size c.

Technique	DFT	SFA	PAA	SAX/iSAX
Time	$O(Nn \log n)$	$O(Nn \log n)$	$O(Nn)$	$O(Nn)$

Table 1: Asymptotic runtime complexity of dimensionality reductions for N approximations of time series of length n.

3.7 Memory Complexity

Claim 3. *Given N SFA words of length w, SFA requires $O(Nw)$ bytes for an alphabet of size* 256.

Proof. The memory footprint of SFA consists of the size of the approximations and the size of the lookup tables. An alphabet of size 256 can be encoded in $\log_2 256$ bits or 1 byte. Thus, an approximation of length w requires w bytes. Given N time series this results in Nw bytes. Additionally SFA requires an overhead of $w(c-1) \cdot 8$ bytes for storing the w lookup tables containing the $c-1$ real-valued breakpoints with 8 bytes (Double) each. Since these lookup tables are stored only once, the overhead in terms of memory is negligible for large N, resulting in a total complexity of $O(Nw)$ bytes. □

Claim 4. *SFA has the same asymptotic memory complexity as iSAX.*

Proof. For an alphabet of size 256 iSAX/SAX requires $w \log_2 256$ bits or w bytes for each approximation. Additionally iSAX requires a small overhead of $(c-1) \cdot 8$ bytes for the lookup table containing the $c-1$ real-valued break points with 8 bytes (Double) each. Following the same reasoning as for SFA, this overhead is negligible for large N, resulting in a total complexity of $O(Nw)$ bytes. □

In case of numerical dimensionality reductions each real-valued coefficient of the approximation is stored in a Double, which allocates 8 bytes, resulting in a total of $w \cdot 8$ bytes for each approximation. Symbolic representations allow for indexing terabyte sized data [26, 5] due to this 8-fold lower memory footprint by utilising discretisation on top of approximation (Table 2).

Technique	Numerical	SFA	SAX/iSAX
Time	$O(Nw8)$	$O(Nw)$	$O(Nw)$

Table 2: Asymptotic memory complexity in bytes of dimensionality reductions for N approximations of length w and an alphabet of size 256 **for symbolic representations SFA and SAX/iSAX.**

4. INDEXING SFA

The purpose of indexing is to partition the search space into equal-sized groups containing similar entries and to

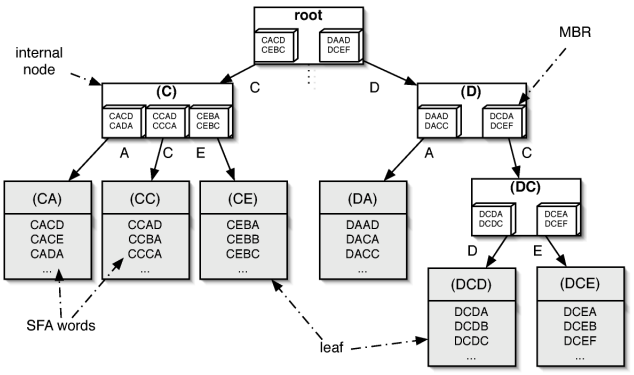


Figure 5: SFA trie: for simplicity the MBRs show SFA symbols instead of Fourier coefficients.

provide a fast lookup for these entries. The idea of indexing SFA is simple: grouping is done based on the first k coefficients (characters) of the SFA words. Increasing k leads to smaller sized groups. As the entries are not equally distributed in the search space, k is individually increased until all entries are equally distributed over each group.

SFA is based on Fourier transformation. A Fourier transformation of a real-valued time series of length n can be represented using $\frac{n}{2}$ Fourier-coefficients, as the later Fourier-coefficients are complex conjugates of the first $\frac{n}{2}$ coefficients according to [24], except for the very first Fourier-coefficient. Each Fourier-coefficient individually contributes to the *global information*. By adding coefficients to a k-sized group, we can add further detail without the need to recalculate the first k-coefficients or the other groups being affected. The first lower frequencies provide more information about the time series (*inherent ordering*). Due to this inherent ordering, we just add the $(k+1)$-th coefficient to a k-sized group for indexing (simple split criterion).

This means that SFA is best computed at a high word length, but only a few groups exploit all symbols of the words.

The inherent ordering and the global information are maintained by the SFA approximation and SFA trie:

- *inherent ordering*: the i-th level of the index equals the i-th character of the SFA word.
- *global information*: splitting a leaf node at the i-th level is performed by adding the $(i+1)$-th character, thus extending the leaf node in depth by one.

In essence, the SFA trie is a prefix tree built over a prefix of the SFA word. A prefix tree is defined as follows:

- *Edges*: Each edge is labelled by a symbol of the alphabet $\Sigma = \{symbol_1, \ldots, symbol_c\}$.
- *Internal Node*: Each node is labelled by a prefix $c_1 \ldots c_i$ that is defined by concatenating the labels on the edges from the root node to that node. An internal node has at most $|\Sigma|$ children. Given an internal node with the prefix $c_1 \ldots c_i$, it's children have the prefixes $c_1 \ldots c_i a$, with $a \epsilon \Sigma$ being the label on the incoming edge.
- *Leaf Node*: Leaf nodes contain words. The leaf node labelled $c_1 \ldots c_i$ contains all words, which share the prefix $c_1 \ldots c_i$. For any key, built over the alphabet Σ, there is at most one leaf node, which is labelled by a prefix of that key.
- *Root Node*: The root node is an internal node, which is labelled by an empty prefix.

In the SFA trie, each node represents the time series sharing the same prefix of an SFA word. The SFA trie has the following modifications/properties:

1. *Controlled Fanout*: Each node has at most $|\Sigma|$ children.
2. *Overlap-free*: Space is partitioned using the SFA alphabet Σ. This guarantees that the trie nodes are overlap-free, which is an important property of an index [2].
3. *Threshold*: A threshold th is used to control the maximal number of time series for each leaf node until it is split. This guarantees that the paths within the trie do not get too sparse.
4. *Split*: A split is performed, after the $(th+1)$-th element is inserted into a leaf node. Splitting a leaf node with prefix length k is performed by relabelling the leaf node as an internal node and simply reinserting the SFA words into this node using a prefix length $k+1$.
5. *Controlled Depth*: The depth of the SFA trie is limited by the SFA word length.
6. *Pointers*: Each leaf node contains a pointer to the original time series stored on the disk.
7. *MBR*: Each internal node V with prefix $c_1 \ldots c_i$ contains a minimum bounding rectangle (MBR), which is a w-dimensional list of tuples $((\min_1, \max_1) \ldots (\min_w, \max_w))$, where w equals the SFA word length. It contains the real-valued upper and lower bounds for each dimension of the DFT approximations contained in all children of V. Given the DFT approximations $TS = \{T'_1, \ldots, T'_n\}$ of all SFA words with prefix $c_1 \ldots c_i$, the a-th tuple $(\min_a, \max_a)$ of an MBR is defined by the minimum and maximum over the a-th dimension:
$\min_a = \min\{t'_a \,|\, T'_j = (t'_o \ldots t'_{w-1}) \epsilon TS\}$ and
$\max_a = \max\{t'_a \,|\, T'_j = (t'_o \ldots t'_{w-1}) \epsilon TS\}$.

The SFA trie is a prefix tree (Figure 5). The trie makes use of MBRs, a threshold and the simple split criterion in combination with the frequency domain. The SFA trie is not height balanced but has a maximal fanout.

The novelty of the trie is that it adapts to dense leaf nodes by increasing the length of the prefix, which equates to improving the quality of the approximation without altering all the other leaf nodes. This increases accuracy of the similarity search, i.e. less leaf nodes have to be searched for answering a similarity query.

4.1 SFA Trie: Operations

4.1.1 Insertion

Given a time series T, it's SFA representation T_{SFA} and the DFT approximation T_{DFT}, we have to find the leaf node, whose label is a prefix of T_{SFA}. Given a node, we retrieve the one child node, which has the same prefix (line 1). Next, we

Algorithm 1 SFA Trie Insertion

function Insert(T,T_{SFA}, T_{DFT}, i, *node*)
1: child = node.getChild($T_{SFA}[i]$)
2: **if** (child != null)
3: child.adaptMBR(T_{DFT})
4: **if** (child.isInternal())
5: Insert($T, T_{SFA}, T_{DFT}, i+1, child$)
6: **else if** (child.isLeaf())
7: child.add(T)
8: **if** (child.split())
9: child.setInternal()
10: **foreach** TS **in** child
11: Insert($TS, TS_{SFA}, TS_{DFT}, i+1, child$)
12: **else**
13: newchild = new Leaf
14: node.add($T_{SFA}[i]$, newchild)
15: newchild.add(T)

adapt the MBR of that child using the DFT approximation (line 3).

If that child is an internal node (line 4), we recursively insert the time series and increase the length of the prefix by one. If that child is a leaf node (line 6), we add the time series and check if the insertion caused the leaf to split (line 8). For splitting, we simply flag the child node as an internal node, and reinsert all time series TS into that node (line 9–11), causing the length of the prefix to grow by one.

If the child node does not exist (line 12–14), we create a new child and add the time series.

4.1.2 Complexity of Insertion

Claim 5. *The total complexity for approximation (SFA) and insertion (SFA trie) of a time series of length n is $O(n \log n + w(w+th))$ operations.*

Proof. The number of operations for insertion is bounded by the depth of the SFA trie, which is limited by the SFA word length w. Thus, descending the SFA trie to find the corresponding leaf node requires $O(w)$ operations, assuming we use a hash for storing (label,edge)-pairs with $O(1)$ lookup (line 1). Adjusting the MBRs (line 3) requires $O(w)$ operations for each level of the trie, due to the need to calculate the minimum and maximum value for each dimension. Splitting (line 10-11) is applied locally and a maximum of th time series are reinserted causing the MBRs to adjust. This leads to a total of $O(w \cdot (w+th))$ operations for insertion in case of a split and $O(w^2)$ otherwise. This results in a total complexity for approximation and insertion of a time series of length n of $O(n \log n + w(w+th))$ operations. □

4.1.3 SFA Trie - MBR distance

During query processing, the minimal distance of a query to an MBR is calculated. An MBR consists of real-valued upper and lower bounds: MBR=$((l_1,u_1),\ldots,(l_w,u_w))$. The distance between a DFT representation $Q_{DFT} = (q'_1 \ldots q'_w)$ and an MBR is defined as a sum over all dimensions similar to SFA (Eq. 3):

$$mindist^2(MBR, Q_{DFT}) = 2\sum_{i=2} dist_{mbr}((l_i,u_i), q'_i)^2$$

The distance between a DFT coefficient q'_i and a tuple (l_i,u_i), is defined as the distance to the lower bound, if q'_i is smaller, and the distance to the upper bound, if q'_i is larger.

Algorithm 2 Exact k-Nearest-Neighbour Search

function exactKNNSearch(Q, Q_{DFT}, k)
1: PriorityQueue queue, Queue result
2: PriorityQueue temp = approximateSearch(Q,Q_{DFT},k)
3: queue.push(root, 0)
4: **while** (!queue.isEmpty()) **do**
5: (minimum, distance) = queue.RemoveMinimum()
6: **for all** time series T in temp **such that** \
$D_{true}(T,Q) \leqslant distance$ **do**
7: temp.remove(T)
8: result.insert(T)
9: **if** |result| = k **return** result
10: **if** minimum is internal node
11: tempDist = temp.get(k-|result|).distance
12: **for all** node in minimum.children **do**
13: nodeDist = $mindist(node.MBR, Q_{DFT})$
14: **if** nodeDist<tempDist
15: queue.push(node,nodeDist)
16: **else if** minimum is leaf node
17: signals=retrieve raw time series from hard disk
18: **for all** time series T in signals
19: temp.push(T,$D_{true}(T,Q)$)
20: **return** result

We modify Eq. 4 by replacing breakpoints β_i by the tuple (l_i,u_i):

$$dist_{mbr}((l_i,u_i),q'_i) \equiv \begin{cases} 0, & \text{if } l_i \leqslant q'_i \leqslant u_i \\ l_i - q'_i, & \text{if } q'_i < l_i \\ q'_i - u_i, & \text{if } q'_i > u_i \end{cases} \quad (9)$$

4.1.4 SFA Trie: k-Nearest-Neighbour Exact Search

We introduce the k-NN exact search algorithm for SFA based on the SFA trie. The algorithm is an adaptation of the multistep 1-NN algorithm introduced in [26]. The algorithm uses a candidate list which is generated by using *mindist* in reduced space and later refined by using the original distance D_{true}. In contrast to the multistep 1-NN algorithm it makes use of an approximate search (line 2) and the pruning of index branches based on that search result (line 14).

Input parameters are the query Q, the DFT approximation of the query Q_{DFT} and k. A priority queue *queue* is used for index navigation and storage of the nodes having the smallest distance in reduced space. Another priority queue *temp* is used to store all raw time series currently known. The queue *result* is used to store the k-NN of our query. A time series is transferred from *temp* to *queue* (line 6-8) as soon as there are no index branches with a smaller distance available (line 6-8).

The algorithm starts by performing an approximate search (line 2) within the SFA trie. The approximate search exploits the fact that the nearest neighbours and the query are likely to have the same SFA word. Therefore, we search the leaf node with the same SFA word as the query and add those time series to *temp* to enable pruning of the trie (line 14).

Starting with the root node of the index (line 3), we pop the node having the smallest distance from *queue* (line 5). If the node is an *internal node* and it's distance is lower than that of the best so far known time series in *temp* (line 13), we traverse and insert all child nodes and their *mindist* into *queue* (line 10-15). The distance of the k-th best so far known time series is used for pruning (line 11), which is the $(k - |result|)$-th element in *temp*. If the node is a *leaf node* (line 16), we retrieve the raw time series from disk (line 17)

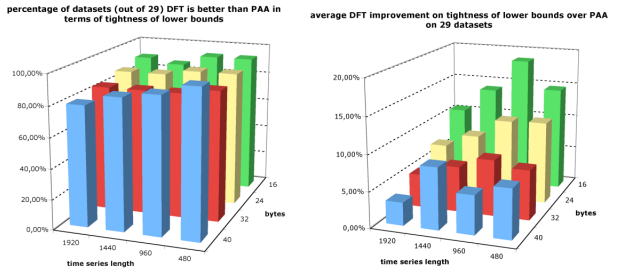


Figure 6: Comparison of DFT and PAA for the 29 datasets published in [26].

and insert them and their true distance to the query into *temp* (line 19). The algorithm terminates if *queue* is empty (line 4) or we have found all k-NN (line 9).

5. EXPERIMENTAL EVALUATION

In our experimental evaluation we show the effectiveness of SFA in terms of pruning power and exact k-NN query performance. We compared our algorithms with numerical and symbolic dimensionality reductions. We implemented all algorithms in Java and the experiments were performed on an AMD Opteron 855 2.6 GHz running SuSE SLES 10 using JRE x86_64 1.6. For all experiments the time series were z-normalised prior to the experiment.

We ran the experiments on 29 datasets presented in [26]. The exact similarity search experiments were additionally performed on 10 synthetic datasets from the UCR data mining archive [14]. We created a website [25] which contains raw numbers and some additional figures.

5.1 Tightness of Lower Bounds

To measure the impact of the dimensionality reduction *DFT* and the discretisation scheme *MCB* as part of *SFA*, we measured the *tightness of lower bounds measurements* (tlb). The tlb is defined as the lower bounding distance over true distance [26].

$$tlb = D_{IndexSpace}(Q', C')/D_{True}(Q, C)$$

'Higher' is 'better' and when the tlb reaches 1, the search in reduced space will return the original time series.

5.1.1 Impact of Dimensionality Reductions

Previous publications showed that there is little difference between dimensionality reductions [21]. However, we looked at the results presented in [26], which are complemented by a web-page documenting the tlb for 29 time series databases.

Surprisingly DFT was better than PAA in at least 79.31% and up to 96.55% of the datasets (Figure 6 left) depending on the number of bytes and the length of the time series.

We furthermore measured the relative difference in the tlb given by $1 - \frac{tlb_{DFT}}{tlb_{SAX}}$ and averaged these results over all datasets (Figure 6 right). Again DFT has a 3.2% to 18.68% better tlb than PAA. While the difference in the tlb may seem small, this results in a $10-30\%$ increase in the pruning of the search space during the query execution for DFT (see Figure 8), i.e. $10-30\%$ less disk accesses.

length	2048	2048	1024	1024	256	256
symb.	256	256	256	256	256	256
coef.	16	8	16	8	16	8
N	28	28	28	28	29	29
W_+	379.5	385	390.5	386.5	415.5	404
p-value	**3.06e-5**	**1.79e-5**	**1.03e-5**	**1.54e-5**	**9.74e-6**	**2.49e-6**

Table 3: Wilcoxon signed-rank test for null-hypothesis $tlb_{DFT} \leq tlb_{PAA}$. The null-hypothesis can be rejected with a p-value of at least 3.063e-05.

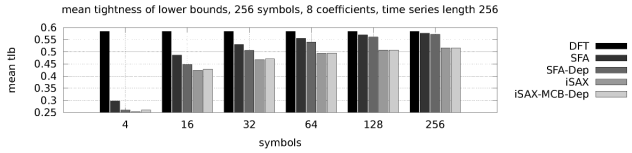


Figure 7: The mean tightness of lower bounds over 29 datasets using different discretisation techniques for a varying number of symbols.

5.1.2 Wilcoxon Signed-Rank Test

To confirm these results, we performed additional tlb experiments on the 29 datasets for different time series lengths and numbers of coefficients (see also our website). We applied a one-tailed Wilcoxon signed-rank test using the statistics tool R to test if any method has a higher tlb. The null-hypothesis is that tlb_{PAA} is higher than tlb_{DFT}:

$$H_0: \quad tlb_{DFT} \leq tlb_{PAA}$$

A null-hypothesis can be rejected if the p-value is smaller than 0.01. With a p-value of at least 3.06e-05 the null-hypothesis H_0 can be rejected on all tested configurations (Table 3). This means, the alternative hypothesis that *"DFT has a higher tlb than PAA"* is statistically significant.

5.1.3 Impact of Discretisation Scheme

To show the impact of MCB discretisation, we compared the influence of discretisation schemes on the *tlb*. We measured the tlb on the 29 datasets for SAX and SFA using different discretisation techniques. *SFA* is using MCB discretisation with equi-depth binning. *SFA-Dep* is using equi-depth binning of the real distribution (without MCB). *iSAX-MCB-Dep* is using PAA, MCB discretisation and equi-depth binning. *iSAX* is using PAA and equi-depth binning of Gaussian distribution. The results (Figure 7) have been averaged over 1.000 random pairwise comparisons on each of the 29 datasets for time series with an original length of 256 using a constant 8 coefficients for the approximations.

DFT without discretisation shows the highest tlb, as any discretisation technique involves loss of information.

iSAX-MCB-Dep does not improve over iSAX, as in the time domain the distribution for each coefficient will be the same for subsequences, due to all points being shifted over all positions of the time series.

SFA shows a very tight *tlb* to DFT for 128 to 256 symbols. As assumed before, the tighter tlb of DFT over PAA (Figure 6) leads to a tighter tlb of SFA over iSAX.

SFA-Dep starts with a rather poor tlb and the difference to SFA becomes small for 256 symbols. This is due to the small average size of the discretisation intervals, i.e. a size

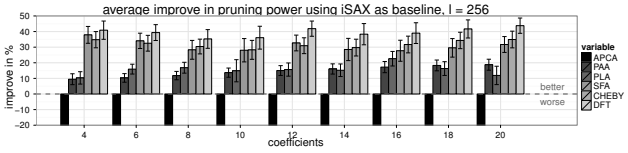


Figure 8: Average pruning power over 29 time series datasets using iSAX as baseline (equals 0%) with varying number of coefficients (4 to 20). Each bar represents the average over 29 datasets.

of $\frac{2}{|symbols|}$ for a value rang of $[-1, 1]$.

We conclude that differences in the discretisation intervals have a smaller impact on the tlb than the approximation when the number of symbols gets large. However, for the SFA trie high precision using small number of symbols is required, e.g. 4–16 symbols. Thus, SFA is favourable over SFA-Dep and iSAX. SFA is favourable over DFT because of its 8-fold smaller memory footprint.

5.2 Pruning Power

Motivated by the high tlb for SFA, we compared SFA with numerical and symbolic dimensionality reductions in terms of pruning power, which is the benchmark for the performance of 1-NN queries. The *pruning power* P is defined as the fraction of the database, that must be examined in reduced space before the 1-NN to a query is found [15]:

$$P = \frac{\text{number of objects to be examined}}{\text{total number of objects}}$$

Pruning Power Benchmark.

In this experiment we compare SFA to the most recently published numerical dimensionality reductions. We use iSAX pruning power as the base line and measure the relative improvement of the other dimensionality reductions given by $1 - \frac{P_{current}}{P_{iSAX}}$ in percent, thus *higher is better*. The results were averaged over 29 datasets for a time series length l of 256, an increasing number of coefficients to represent one approximation and 256 symbols for SFA and iSAX.

Figure 8 shows that among the numerical representations DFT shows the best pruning power. SFA is better than most of the other numerical representations and slightly worse than DFT. SFA is on average up to a factor of 0.40 better than iSAX, that means on average 40% less time series have to be inspected to find the 1-NN.

Overall SFA is not only better than iSAX but the results indicate that it is competitive with the best numerical reductions. We conclude that these improvements are a result of the tight tlb achieved by MCB and DFT. On our website [25] we show the pruning power for each of the 29 datasets.

DFT has the best pruning power. However, the advantage of SFA over DFT is the symbolic representation and thus 8-fold lower memory footprint. This allows for indexing large and high dimensional datasets through the SFA trie. This is what we will show in the next experiments.

5.3 Indexing High Dimensional Datasets

In these experiments we focus on two parameters:

1. Approximation length/word length: this represents the indexable dimensionality by the multidimensional index structures.

2. Dimensionality of the datasets: this represents the length of each raw time series.

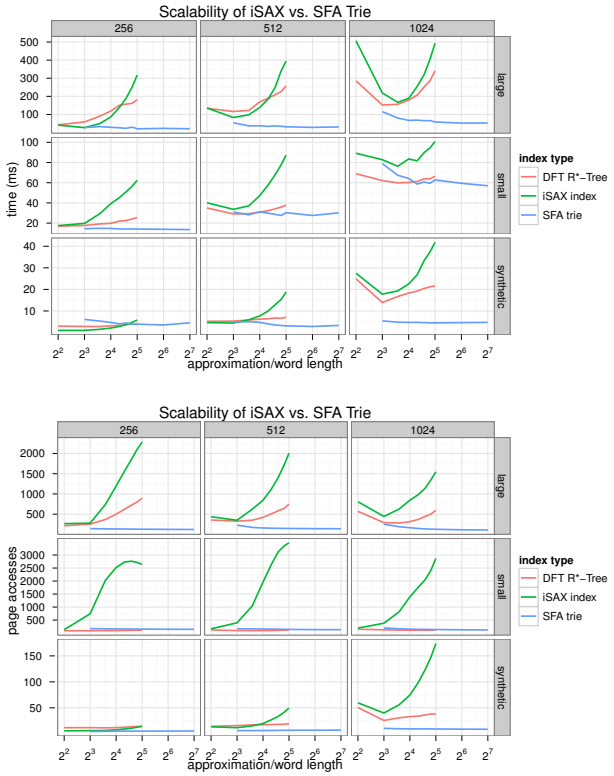


Figure 9: Mean wall clock time and page accesses for real world and synthetic datasets.

Datasets.

We evaluate the performance of similarity search using SFA, iSAX and DFT on 10 synthetic random walk datasets from [14] and the real world datasets presented in [26]. We chose DFT as a representative for the numerical reductions, as DFT was best in the tlb and pruning power experiments. The synthetic datasets contain 100,000 overlapping time series each. The real world datasets contain 2.500 to 1 million overlapping time series. For the experiments we partitioned the real world datasets into two groups *small* and *large*, with 16 real datasets consisting of less than 100,000 time series and 11 real datasets consisting of more than 100,000 time series. We excluded *npo141* and *cl2fullLarge* from the large datasets, as their wall clock time and page accesses are so high that they would dominate the other measurements.

Setup.

We compare the SFA trie with the iSAX index and DFT using R*-trees [3], a variant of the R-tree [13]. We set the base cardinality (alphabet size) $b = 4$ and the threshold $th = 100$ for the iSAX index and the $fillfactor = 100$ and $p = 0.80$ for the R*-tree. We set the fanout of the SFA

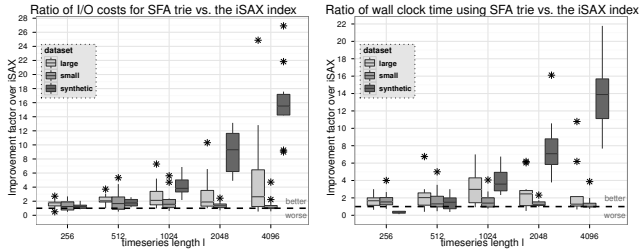


Figure 10: Ratio of page accesses and wall clock time for the SFA trie vs. the iSAX index on the 37 datasets.

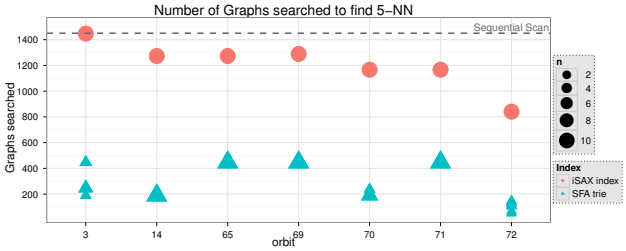


Figure 11: Number of graphs searched to find the 5-NN out of 1450 graphs. Each dot represents 1 to 10 queries. Only orbits with $k < 10^6$ are shown.

trie to 8 (equals 8 symbols) and the threshold $th = 100$. We tested using 256 symbols for iSAX. Each index has a maximum of 100 time series per leaf (same as in [26]). All results have been averaged over 1000 10-NN queries.

A leaf node access accounts for one page access even if the leaf node spans more than one disk block or if two leaf nodes are stored in successive blocks on hard disk.

Impact of Approximation/Word Length.

In this experiment we test the scalability of the SFA trie in terms of the length of the approximations, i.e. varying the word length, which correlates with the indexable time series length. This experiment was performed on large, small and synthetic datasets. Figure 9 shows the scalability in terms of the average wall clock time and the average page accesses to answer one k-NN query for an increasing length of the approximations. The minimum of each line shows the best configuration for each index structure. The labels on the top represent the time series lengths l. The labels on the right represent the different datasets.

Time Series Lengths (left to right plots): The iSAX index degenerates faster than the SFA trie when increasing l from 256 to 1024. The R*-tree is competitive to the iSAX index on the small datasets but scales badly with increasing l on the large and synthetic datasets.

X-Axis (Indexed Dimensions): Both measurements (wall clock time and page accesses) show the same trend. The SFA trie scales to 128 indexed dimensions without degeneration. $20 - 32$ dimensions seem to be a good trade-off between storage space and performance. This equals $60 - 96$ Bits for SFA. Comparing the optimal number of indexed dimensions for iSAX (4−8), DFT (8−16) and SFA (20−32), the SFA trie is among the best indexes on the small and large real world datasets and only scores worse on the synthetic datasets for $l = 256$. iSAX performs well on random data, because the iSAX discretisation is based on normal distribution.

In the SFA trie an increasing dimensionality of the approximations causes dense leaf nodes to split, but the general structure of the trie remains unaffected as opposed to the R*-tree or the iSAX index. This is why the SFA trie improves when increasing the length of the approximation.

Indexing High Dimensional Datasets.

For this experiment we compare the symbolic representations in terms of indexing time series datasets with a high dimensionality (very long time series). Only the optimal number of coefficients, i.e. those which had the lowest data page accesses, is compared for each dataset and representation. In Figure 10 we illustrate the factor of improvement in wall clock time and page accesses for the SFA trie over the iSAX index for different time series lengths l. The factor is given by $\frac{time_{iSAX}}{time_{SFA}}$ and $\frac{io_{iSAX}}{io_{SFA}}$, meaning *higher is better.*

The largest dataset with 4096 dimensions and size 10^6 occupied approx. 32GB of memory.

On the large datasets the median improvement factor over iSAX rises with increasing l from about 1.5 to 3 (time) and to 2.8 (I/O). On some datasets the improvement is up to a factor of 25 (11) in terms of I/O (time) for real (*converted-katafu*) and 27 (22) in terms of I/O (time) for synthetic datasets (*synthetic 9*). On the small datasets there seems to be little influence of l on the wall clock time and the median stays around a factor of 1.5, with some datasets showing an improvement of up to 5 (*foetal-ecg/power-data*). On the synthetic datasets and $l = 256$ the SFA trie shows a worse wall clock time than the iSAX index. However, the median improves to a factor of about 1.5 to 15.

We conclude that the SFA trie scales much better for increasing time series lengths on large and synthetic datasets due to the high number of indexable dimensions, as shown in the previous experiment.

5.3.1 *Use-Case: Cellular Networks*

This experiment illustrates the necessity of indexing time series with a length of up to 66000 dimensions. Comparing cellular networks is a challenging problem that could provide insight into biology and therapeutics. Thus, heuristics such as graphlet frequency distribution have been proposed. In [23] two networks are similar, if they contain similar subgraphs, named *graphlets*. A network is characterised by up to 73 graphlet degree distributions *GDD* of graphlets with 2–5 nodes. The GDD measures the number of nodes $N_G^j(k)$ touching k graphlets at a particular node j in the graphlets, called orbit. For a fixed orbit j the GDD can be easily interpreted as a time series using k as the time axis and $N_G^j(k)$ as the value. However, k might become arbitrarily large. The similarity D^j of two graphs G and H is measured using the j-th normalised GDD:

$D^j(G,H) = (\sum_{k=1}^{\infty} \left[N_G^j(k) - N_H^j(k)\right]^2)^{1/2}$, $j \epsilon [0 \dots 72]$

We indexed different orbits for 1450 randomly generated graphs from [10], consisting of 725 isomorph pairs, containing 20–600 nodes with eta=0.01. We indexed those orbits with a k degree of less than 66000 using the SFA trie and the iSAX index and did a 5-NN search, which is guaranteed to find the isomorph pair. Figure 11 shows that, due to the high degree k of the GDD, the iSAX index falls back

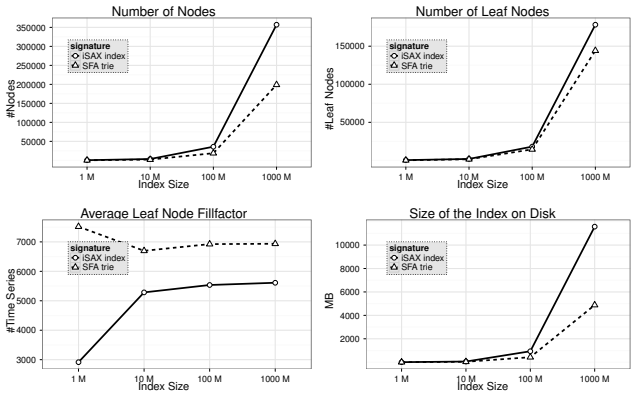


Figure 12: Comparison of index size and leaf node occupancy with varying size of the synthetic dataset.

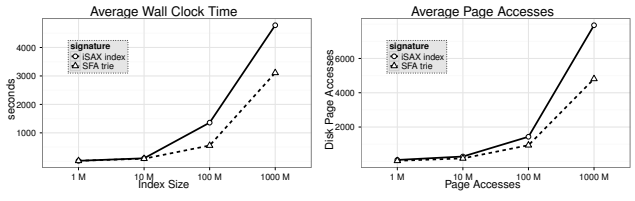


Figure 13: Average wall clock time and average page accesses to answer a 5-NN query with a varying size of the synthetic datasets.

to a sequential scan of all graphs. Meanwhile, the SFA trie, which was built using an SFA word length of 1024 up to 2048, scales well with the high degree k of the GDD and on average 150–400 out of 1450 graphs are scanned to find the 5-NN for an orbit. This is a factor of 3–7 improvement compared to iSAX and sequential scanning.

5.4 Indexing Large Datasets (One Billion Time Series)

In this experiment we focus on the size of the dataset, i.e. the number of time series it contains.

Dataset.

iSAX and SFA qualify for indexing large sized datasets as both are symbolic representations and exhibit an approx. 8-fold lower memory footprint than numerical dimensionality reductions. For this experiment, we indexed 10^6 to 10^9 synthetic time series with a length of 256, resulting in 2GB to 2TB raw data. Each data point was generated using standard normal distribution N(0,1) and the recursive function: $x_{i+1} = N(x_i, 1)$ (same as in [5]).

Setup.

We set the word length $w = 8$ and base cardinality $b = 4$ for the iSAX index and $w = 20$ for the SFA trie and a leaf threshold $th = 10000$ for both. The iSAX index was implemented using the optimised node splitting policy presented in [5].

A leaf node access accounts for one page access even if the leaf node spans more than one disk block or if two leaf nodes are stored in successive blocks on hard disk.

Index Size & Leaf Node Occupancy.

In this experiment we measure the size of the index in (1) the total number of nodes (including leaf nodes), (2) the total number of leaf nodes, (3) the fillfactor of the leaf nodes and (4) the size of the index on disk excluding the raw time series data. Figure 12 shows that the SFA trie is a more compact representation than the iSAX index and requires a factor of 1.72 to 2.28 less nodes and 1.24 to 2.58 less leaf nodes to represent the data. This leads to a 1.24 to 2.58 higher fillfactor of the leaf nodes. The SFA trie is also more compact in terms of memory requirements, as it needs up to a factor of 2.4 less main memory compared to the iSAX index (excluding the raw time series data).

These results show that in terms of memory footprint the SFA trie qualifies for indexing large datasets such as the iSAX trie does [5, 26].

Similarity Search.

To benchmark the index structures, we compared the iSAX index and the SFA trie in terms of leaf node accesses, which is an objective measure for disk page accesses, and wall clock time, which is the time spent for each similarity query including disk page accesses. We did 100 5-NN queries each and averaged the results.

This experiment (Figure 13) demonstrates that the SFA trie requires less disk accesses and less wall time than the iSAX index to answer a similarity query. On average the SFA trie requires up to a factor of 2.45 less wall time and up to a factor of 2.6 less leaf node accesses.

A 5-NN query on the 1000 M dataset took on average 52 minutes for the SFA trie and 80 minutes for the iSAX index.

Note that the use of N(0,1) for data generation is in favour of iSAX, as iSAX discretisation is based on N(0,1). Still, the SFA trie requires less page accesses and wall time than the iSAX index.

6. CONCLUSIONS

We introduced the SFA trie and the symbolic representation SFA based on the discretisation of Discrete Fourier Transform (DFT). The SFA trie exploits the frequency domain nature of SFA by approximation of a time series using a high dimensionality and a variable prefix for indexing. With a variable prefix length it is possible to distinguish time series which have similar approximations. This leads to improved similarity query performance. The SFA trie is tailored for a variable prefix length as it grows in depth rather than width when increasing the length of similar approximations, which postpones the effects of the Curse of Dimensionality.

As part of SFA we introduced a novel discretisation technique called MCB. As DFT has a statistically significant tighter lower bound compared to PAA, SFA provides a tighter lower bound to the Euclidean distance than iSAX. Like iSAX it offers a high information density per bit and thus allows for indexing large datasets. Unlike iSAX, every SFA coefficient represents the whole signal due to the frequency domain, which allows for indexing high dimensional datasets. In our experiments the SFA trie is the best index structure in terms of page accesses and wall clock time on real and synthetic datasets. Future work includes a technique to efficiently rebalance the SFA trie, thereby avoiding degeneration due to updates.

Acknowledgement

The authors would like to thank Eamonn Keogh, for his valuable comments on previous versions of the paper and making available the time series datasets, and Ulrike Golas, for her valuable comments on the final version of the paper.

7. REFERENCES

[1] Agrawal, R., Faloutsos, C., and Swami, A. N. Efficient Similarity Search In Sequence Databases. In *Proc. (FODO)* (1993), Springer Verlag, pp. 69–84.

[2] Assent, I., Krieger, R., Afschari, F., and Seidl, T. The ts-tree: efficient time series search and retrieval. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology* (New York, NY, USA, 2008), EDBT '08, ACM, pp. 252–263.

[3] Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. The R*-tree: an efficient and robust access method for points and rectangles. *SIGMOD Rec. 19*, 2 (1990), 322–331.

[4] Cai, Y., and Ng, R. Indexing spatio-temporal trajectories with chebyshev polynomials. In *Proc of the 2004 ACM SIGMOD* (2004), ACM, pp. 599–610.

[5] Camerra, A., Palpanas, T., Shieh, J., and Keogh, E. isax 2.0: Indexing and mining one billion time series. *Data Mining, IEEE International Conference on 0* (2010), 58–67.

[6] Chakrabarti, K., Keogh, E., Mehrotra, S., and Pazzani, M. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proc. of ACM SIGMOD* (2002), pp. 151–162.

[7] Chan, F. K.-P., chee Fu, A. W., and Yu, C. Haar wavelets for efficient similarity search of time-series: With and without time warping. *IEEE Transactions on Knowledge and Data Engineering 15*, 3 (2003), 686–705.

[8] Chan, K.-P., and chee Fu, A. W. Efficient time series matching by wavelets. In *ICDE* (1999), pp. 126–133.

[9] Chen, Q., Chen, L., Lian, X., Liu, Y., and Yu, J. X. Indexable PLA for efficient similarity search. In *Proc of the VLDB* (2007), pp. 435–446.

[10] De Santo, M., Foggia, P., Sansone, C., and Vento, M. A large database of graphs and its use for benchmarking graph isomorphism algorithms. *Pattern Recogn. Lett. 24* (May 2003), 1067–1079.

[11] Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. Fast subsequence matching in time-series databases. In *SIGMOD Rec.* (1994), pp. 419–429.

[12] Gaede, V., and Günther, O. Multidimensional access methods. *ACM Comput. Surv. 30*, 2 (1998), 170–231.

[13] Guttman, A. R-trees: a dynamic index structure for spatial searching. In *Proc of the 1984 ACM SIGMOD* (1984), ACM, pp. 47–57.

[14] Keogh, E. The UCR time series data mining archive. http://www.cs.ucr.edu/ eamonn/TSDMA/index.html, 2002.

[15] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems 3*, 3 (2001), 263–286.

[16] Keogh, E., and Pazzani, M. A simple dimensionality reduction technique for fast similarity search in large time series databases. In *Knowledge Discovery and Data Mining PAKDD 2000* (2000), vol. 1805, Springer, pp. 122–133.

[17] Korn, F., Jagadish, H. V., and Faloutsos, C. Efficiently supporting ad hoc queries in large datasets of time sequences. In *Proc of the 1997 ACM SIGMOD* (1997), ACM, pp. 289–300.

[18] Lin, J., Keogh, E., Lonardi, S., and chi Chiu, B. Y. A symbolic representation of time series, with implications for streaming algorithms. In *DMKD* (2003), pp. 2–11.

[19] Lin, J., Keogh, E., Wei, L., and Lonardi, S. Experiencing SAX: a novel symbolic representation of time series. *Data Min. Knowl. Discov. 15*, 2 (2007), 107–144.

[20] Marwan, N., Thiel, M., and Nowaczyk, N. R. Cross recurrence plot based synchronization of time series. *Nonlinear Processes in Geophysics 9*, 3/4 (2002), 325–331.

[21] Palpanas, T., Vlachos, M., Keogh, E., Gunopulos, D., and Truppel, W. Online amnesic approximation of streaming time series. In *ICDE* (2004), pp. 338–349.

[22] Popivanov, I. Similarity search over time series data using wavelets. In *ICDE* (2002), pp. 212–221.

[23] Pržulj, N. Biological network comparison using graphlet degree distribution. *Bioinformatics 26* (March 2010), 853–854.

[24] Rafiei, D., and Mendelzon, A. Efficient retrieval of similar time sequences using DFT. In *Proc. FODO Conference, Kobe* (1998), pp. 249–257.

[25] Schäfer, P., and Högqvist, M. SFA web page. http://www.zib.de/patrick.schaefer/sfa/, 2011.

[26] Shieh, J., and Keogh, E. iSAX: indexing and mining terabyte sized time series. In *KDD* (2008), pp. 623–631.