Window Size Selection In Unsupervised Time Series Analytics: A Review and Benchmark

Arik Ermshaus[1], Patrick Schäfer[1], and Ulf Leser[1]

Humboldt-Universität zu Berlin, Germany
{ermshaua,patrick.schaefer,leser}@informatik.hu-berlin.de

Abstract. Time series (TS) are sequences of values ordered in time. Such TS have in common, that important insights from the data can be drawn by inspecting local substructures, and not the recordings as a whole. ECG recordings, for instance, are characterized by normal or anomalous heartbeats that repeat themselves often within a longer TS. As such, many state-of-the-art time series data mining (TSDM) methods characterize TS by inspecting local substructures. The window size for extracting such subsequences is a crucial hyper-parameter, and setting an inappropriate value results in poor TSDM results. Finding the *optimal* window size has remained to be one of the most challenging tasks in TSDM domains, where no domain-agnostic method is known for learning the window size. We provide, for the first time, a systematic survey and experimental study of 6 TS window size selection (WSS) algorithms on three diverse TSDM tasks, namely anomaly detection, segmentation and motif discovery, using state-of-the art TSDM algorithms and benchmarks. We found that WSS methods are competitive with or even surpass human annotations, if an interesting or anomalous pattern can be attributed to (changes in) the period. That is because current WSS methods aim at finding the period length of data sets. This assumption is mostly true for segmentation or anomaly detection, by definition. In the case of motif discovery, however, the results were mixed. Motifs can be independent of a period, but repeat themselves unusually often. In this domain, WSS fails and more research is needed.

Keywords: Time Series, Unsupervised, Learning, Subsequences, Windows, Data Mining, Motif Discovery, Segmentation, Anomaly Detection, Review, Benchmark

1 Introduction

Time series (TS) are sequences of real values ordered in time. Use cases include biological processes like ECG recordings of patient's heartbeats, EEG recordings collected from people taking a nap or physical processes like printer pressure pump sensor recordings from industrial printers. Such recordings have in common, that the main insights from the data can be drawn by inspecting local substructures, and not the recordings as a whole. As such, most state-of-the-art TS data mining (TSDM) algorithms are applied to local substructures, which are referred to as *windows*. However, setting the *right* window size is crucial for most of these tasks. When applied to production-ready systems, failure to provide an *optimal* window size may cause bad decision-making and thus monetary damage.

Many state-of-the-art unsupervised TSDM algorithms can be formulated as a self-join of all pairs of subsequences for a given window length. Examples include Matrix Profile

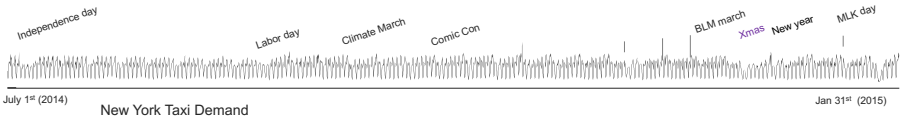

Fig. 1. The famous NYC taxi traffic data set contains anomalous as well as reoccurring local temporal patterns that can be analysed with TSDM techniques.

I-XXV [20], ClaSP [14], EMMA [11], etc. With a poor choice of window length, these methods fail to give good results. While in a supervised setting, the window length can be learned by maximizing an evaluation metric, in an unsupervised setting, such as anomaly detection, segmentation or motif discovery, this is not possible by definition, and algorithms rely on human annotations. Using domain-agnostic methods to learn the window length for unsupervised TSDM is at the core of this experimental study.

Many window size selection (WSS) methods for finding the period length of a TS have been published. This is the first survey and experimental evaluation of WSS for TSDM tasks. We chose three common, yet diverse unsupervised TSDM tasks, namely motif discovery, anomaly detection and segmentation for analysis. We use the famous NYC taxi data set in Figure 1 as an example to illustrate the different requirements of the TSDM tasks. It shows the hourly traffic in NYC throughout the year in 2015. It has a weakly pattern with increasing traffic throughout the weekdays and during rush hours. It has some known anomalies that can be attributed to events, marches or public holidays. The task of anomaly detection aims at finding anomalous structure in a periodic TS [1]. In the NYC taxi data set, this corresponds to anomalous (increased or decreased) traffic at Independence Day, Labor Day, etc. Such anomalies can be attributed to a deviation from the periodicity of the TS. The window size, used for anomaly detection, should thus roughly capture the length of an anomalous event, i.e. some hours or at most a day, or we may miss this anomalous event. The task of segmentation is the process of dividing a periodic TS into disjoint segments, e.g. based on detecting changes in periodicity [6]. In our example, we see a clear weekly or monthly pattern of taxi traffic that repeats throughout the year. These segments themselves are highly repetitive. A window size should thus roughly capture the weekly or monthly periodicity to identify relevant segments. The task of motif discovery aims at finding frequent, similar local patterns in a TS [11]. In our running example, this could be similar traffic on public holidays or on some weekdays, e.g. the first Monday in every month is similar. Other than in the former two use cases, the TS itself need not be periodic, but the window size should be set to discover repetitive local behaviour.

To sum up, the tasks have very diverse requirements for a WSS method: small vs large window sizes, associated with vs independent of the period. The survey and experimental evaluation of WSS techniques is the main focus of this study. The contributions of this paper are as follows:

1. We review 6 established and recent domain-agnostic WSS algorithms from the literature, including technical descriptions, computational complexity analyses and examples for illustration.
2. We benchmarked, for the first time, 6 WSS methods and compared these to human annotations on three diverse and challenging unsupervised TSDM tasks: anomaly detection,

segmentation and motif discovery. For all of these tasks, we tested the state-of-the-art algorithms in their field in combination with all WSS methods on established benchmarks.
3. Overall, we find that if the analysed TS were periodic, as is the case for anomaly detection or segmentation, the WSS methods even surpassed human annotations. If the analysed TS were non-periodic, as for motif discovery, WSS fails.
4. In this work, we make a special effort to provide the sources codes and results on our website [15] to foster follow-up works and reproducibility. We provide a Python implementation of the used methods as well as Jupyter-Notebooks, visualizations and raw measurement sheets.

The remainder of this paper is structured as follows: Section 2 introduces related work, Section 3 reviews the WSS algorithms, Section 4 presents our experimental evaluation, and Section 5 explains our results and concludes the paper.

2 Background and Related Work

In this study, we assume that a TS is generated by observing some output of a physical process, that consists of one (or several) distinct states at arbitrary and a-priori unknown points in time, leading to different measurements [6] in the signal. Unsupervised TSDM algorithms are concerned with finding anomalous subsequences in such signals, detecting their segmentation or instances of similar substructures. We introduce these concepts and the related literature in the following subsections.

2.1 Definitions

Definition 1. *A* time series (TS) T *is a sequence of* $n \in \mathbb{N}$ *real values,* $T = (t_1,...,t_n), t_i \in \mathbb{R}$ *that measures an observable output of a process. The values are also called data points.*

Definition 2. *Given a TS* T*, a* subsequence *(or* window*)* $T_{s,e}$ *of* T *with start offset* s *and end offset* e *consists of the contiguous values of* T *from position* s *to position* e*, i.e.,* $T_{s,e} = (t_s,\ldots,t_e)$ *with* $1 \leq s \leq e \leq n$*. The length (or width) of* $T_{s,e}$ *is* $\|T_{s,e}\| = e - s + 1$.

Definition 3. *A* periodic *TS is one that approximately repeats a subsequence of values after a fixed length of time. We call such a subsequence a temporal pattern (or period) of a TS.*

The two tasks anomaly detection (Section 2.2) and segmentation (Section 2.3) are applied to periodic TS by definition. Still, local parts of a TS can nevertheless deviate from each other, e.g. in period length, shape or amplitude. This is typically due to anomalies, motifs, evolving patterns, or regime changes. In motif discovery (Section 2.4), data need not be periodic, however, but the pattern to be identified repeats unusually often.

2.2 Anomaly Detection

Time series anomaly detection (TSAD) aims to find erroneous or novel behaviour within expected observations. Such outliers are typically categorized as point or subsequence outliers. While the former is an unexpected high (low) data point, the latter is a deviation of the baseline

temporal pattern (as consecutive observations). Blázquez-García et al. [1] formalize the TSAD problem and review techniques to tackle it. Model-based approaches like Isolation Forest (IF) and one-class support vector machine (SVM) learn the temporal dynamics from subsequences of a baseline TS to estimate the novelty of new incoming data points as a reconstruction score. Discord detection is another approach to anomaly detection that reports the subsequence in a TS with the highest pairwise distance to its nearest neighbour as an outlier [20]. It considers subsequences anomalous or unusual in relation to the TS as a reference. Discord detection can be numerically solved as a minimization problem with the matrix profile (MP) [20]. Runtime optimizations exist based on admissible pruning. Frequentist methods further measure the novelty of subsequences by the count of their occurrence, and related information-based algorithms utilize their rate of compressibility as a means of uniqueness.

2.3 Segmentation

The time series segmentation (TSS) task assumes that a TS consists of segments that capture the inherent statistical properties and temporal patterns of the process states at hand. The a priori unknown change points (CPs) between segments are assumed to be abrupt shifts of these properties and indicate state transitions [6]. The TSS problem is to partition a TS into disjoint segments, separated by CPs, corresponding to states of the data-generating process. A number of domain-specific TSS methods that can detect changes in TS with suitable value distributions (e.g. piecewise-constant or Gaussian) have been published in the last decades. Truong et al. [17] present a review of such techniques. They compare methods regarding their cost function, search method, and whether the number of change points is known a priori. Search methods like window-based segmentation (Window) [17] then use such cost functions to find meaningful segmentations that minimize the associated costs. More recently, two domain-agnostic subsequence-based techniques have been proposed, namely FLOSS [6] and ClaSP [14, 5]. Both methods use a k-NN to relate similar subsequences and compute a profile, from which CPs can be extracted as local minima (maxima).

2.4 Motif Discovery

Motif discovery in TS (TSMD) has been researched intensively for approximately 20 years [11]. A majority of research is centred on pair motifs, which are defined as the pair of most similar subsequences in a TS. However, substructures commonly do not only occur in pairs but frequently, such as heartbeats in an ECG recording. Thus, it is more natural to think of a motif as a set of frequently appearing and similar subsequences. We will thus focus on those approaches to motif discovery, that aim for finding sets of the most similar subsequences, aka motif sets. EMMA [11] was the first motif set discovery algorithm. It is based on the discretization of subsequences using Symbolic Aggregate approXimation (SAX) [10]. It hashes the discretized SAX words into buckets, where similar subsequences hash into similar buckets, and the buckets are subsequently post-processed to obtain the final motif sets. The concept of Learning Motif (LM) was introduced by Grabocka et al. [7] to better deal with noisy TS. The paper approaches LM discovery as a process which, starting from a random initialization, iteratively modifies a motif core S' to increase its frequency, i.e., the size of the surrounding motif, while keeping its radius fixed. As the frequency function is not differentiable, they propose a smooth Gaussian-kernel approximation that allows using gradient ascent to find

Method Name	Type	Properties	Complexity
DFT	Whole Series	Frequency-Domain	$\mathcal{O}(n\log n)$
AC	Whole Series	Time-Domain	$\mathcal{O}(n\log n)$
AutoPeriod [18]	Whole Series	Hybrid	$\mathcal{O}(n\log n)$
RobustPeriod [19]	Whole Series	Hybrid	$\mathcal{O}(n^2)$
MWF [8]	Subsequence	Summary Statistics	$\mathcal{O}(m \cdot n)$
SuSS [5]	Subsequence	Summary Statistics	$\mathcal{O}(n\log w)$

Table 1. Properties of WSS algorithms.

the hopefully best hidden motif cores. The LM solution is a heuristic, as the optimization problem is non-convex and the gradient ascent might get stuck in a local optimum.

3 Window Size Selection

Window size selection (WSS) algorithms can be divided into two major categories: (a) whole-series-based and (b) subsequence-based. Whole-series-based methods analyse global properties of a signal in order to detect dominant period sizes. They can further be divided into frequency-based and time-based approaches. While the former decompose a TS into single frequency components, the latter measure the self-similarity of the entire signal when shifted by an offset (lag). A dominant frequency or a high correlation for a certain shift is then used to derive an appropriate window size [4]. Subsequence-based methods, in turn, extract local features from TS. They compare how well statistics computed over temporal patterns (with increasing size) align with the global properties of the signal. A small subsequence width in line with the global TS properties is then chosen as a window size [8, 5]. While whole-series-based methods can easily identify strong and repetitive periods in signals, subsequence-based methods are more able to identify window sizes in more diverse TS like with regime changes (different segments) or varying temporal patterns, as they rely on the data set's descriptive statistics as a proxy instead of exact frequencies or shifts.

In the following subsections, we discuss four whole-series-based methods, namely Dominant Fourier Frequency, Autocorrelation, AutoPeriod and RobustPeriod, as well as two subsequence-based algorithms, Multi-Window-Finder and Summary Statistics Subsequence.

3.1 Dominant Fourier Frequency

The Fourier transform decomposes a signal into a sum of sinusoid waves, also called Fourier coefficients. Each of these coefficients is characterized by its frequency, equal to a period length in the TS, and its phase shift. The main hypothesis of this whole-series-based method is that the most dominant sinusoid wave, i.e. with the largest magnitude, in a signal captures its period best.

Definition 4. *The Discrete Fourier Transformation (DFT) of a TS T, $|T| = n$ is a series of complex coefficients $\mathcal{C} := (c_0,...,c_{n-1}) \in \mathbb{C}^n$, such that $c_k := \sum_{j=1}^{n} T_j \cdot e^{-2\pi i \cdot \frac{jk}{n}}$, for $k = 0,...,(n-1)$ and $i = \sqrt{-1}$. The corresponding series of frequencies is defined as $\mathcal{F} = (f_0,...,f_{\lceil \frac{n-1}{2} \rceil})$, such that $f_k := \frac{k}{n}$, for $k = 0,...,\lceil \frac{n-1}{2} \rceil$.*

Algorithm 1 Most dominant Fourier Frequency

1: **procedure** DOMINANTFOURIERFREQUENCY(T)
2: $\mathcal{C}$,$\mathcal{F}$ $\leftarrow$ DFT(T)
3: $\mathcal{M}$,$\mathcal{W}$ $\leftarrow \sqrt{Re(\mathcal{C})^2+Im(\mathcal{C})^2}, \frac{1}{\mathcal{F}}$
4: **return** $\mathcal{W}$[ARGMAX($\mathcal{M}$)]
5: **end procedure**

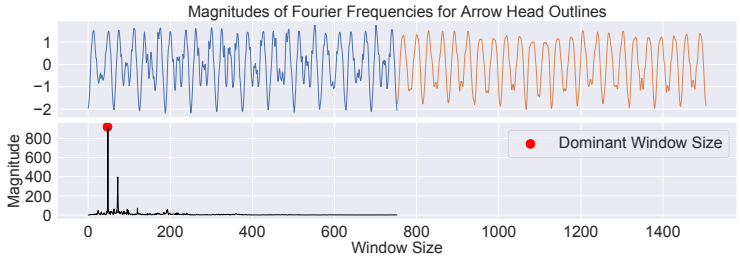

Fig. 2. The magnitudes of the Fourier coefficients show a substantial global maximum for the window size 49 and a smaller local maximum for the window size 72.

A Fourier coefficient c_k describes the phase and magnitude for its associated frequency f_k, with the first c_0 being the mean of the TS T. In order to obtain the magnitude of c_k, we can calculate its modulus $m_k := \sqrt{Re(c_k)^2+Im(c_k)^2}$. Note that we can only capture $\lceil \frac{n}{2} \rceil$ frequencies because of the Nyquist–Shannon sampling theorem.

Algorithmically (see Algorithm 1), we first transform a TS using the Discrete Fourier Transform (DFT), and then select the most dominant Fourier coefficient, from which we may infer the frequency. Its corresponding period size can then be used as a window size. The quality of this computation can be negatively affected by spectral leakage in the DFT as well as large (or multiple) dominant periodicities in the signal. However, windowing techniques can be applied to increase the accuracy of the harmonic analysis [18].

The runtime complexity depends on the computation of the DFT, which is in $\mathcal{O}(n \log n)$. The calculation of the magnitudes and window sizes can be performed in $\mathcal{O}(n)$ as well as the linear search for the most dominant frequency. Thus, the overall runtime complexity is in $\mathcal{O}(n \log n)$. Its space complexity is in $\mathcal{O}(n)$ as it uses four additional arrays.

Figure 2 illustrates instances of different arrowhead shapes as blue and orange segments [2]. The corresponding magnitude profile (Figure 2, bottom) shows a clear global maximum (window size 49, magnitude 921) which captures the period of the signal (between 40-60 data points). The 2nd highest local maximum (window size 72, magnitude 398) also protrudes and captures multiple longer periods containing 70-90 data points.

3.2 Highest Autocorrelation

The autocorrelation (AC) of a TS reports the correlation with a delayed copy of itself. For different time shifts, called lags, it shows how similar the signal is to its shifted version. If the TS has a distinctive period, its AC will be high for the lag that equals the size of the repeated temporal pattern, which in turn can be located and used as a window size.

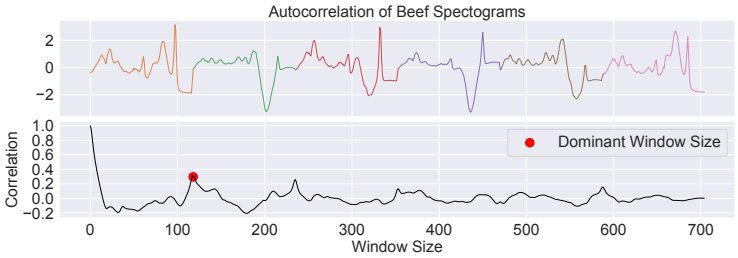

Fig. 3. The AC profile shows substantial deflections for window size 119 and its multiple 236.

Definition 5. *The Autocorrelation Function (ACF) of a TS T, $|T| = n$ defines the zero-normalized cross-correlation $a(l) := \frac{1}{n-l-1} \sum_{j=l+1}^{n} \frac{(T_j - \mu_T) \cdot (T_{j-l} - \mu_T)}{\sigma_T^2}$ for a given lag l and with μ_T,σ_T being the mean or rather the standard deviation of T. The series of cross-correlations $\mathcal{A} := (a(0),...,a(n-1)) \in \mathbb{R}^n$, is the autocorrelation (AC) of T.*

The cross-correlation $a(l)$ captures the similarity of T with its shifted version for a lag of size l. The AC $\mathcal{A}$ of T contains such similarities for all possible lags $0 \leq l \leq n-1$ as a profile. It is highest for no lag and contains local maxima for dominant periods in T as well as their multiples. To determine the period (and hence a window size) for a given TS T, the algorithm calculates the AC $\mathcal{A}$ of T. It then searches for the highest non-trivial local maximum in $\mathcal{A}$ with a peak finding algorithm. Lastly, it reports the lag for the highest peak. While the AC of a signal provides higher quality periodicity estimates, compared to its DFT [18], the selection of the best period size (and not one of its multiples) is more complicated.

The runtime complexity is dominated by the computation of the cross-correlations, which can be reformulated using the Wiener-Khinchin theorem and solved using the FFT in $\mathcal{O}(n \log n)$. The runtime for peak finding using simple baseline approaches utilizing exclusion zones, differencing or shape constraints is in $\mathcal{O}(n)$. Thus, the total runtime is in $\mathcal{O}(n \log n)$. Its space complexity is in $\mathcal{O}(n)$, as it uses one additional array.

As an example of AC based WSS, see Figure 3. At the top, it shows the concatenation of six beef spectrograms [2] (each consisting of 118 data points) and in the bottom we plotted the corresponding AC profile. It shows the trivial global maximum (for no lag) as well as a distinctive local maximum for a lag of 119 (29.6% correlation) and a more subtle one for a lag of 236 (23.5% correlation), which is an approximate 2x multiple of 119. Its substantial local maximum accurately captures a single instance of a spectrogram, which is the dominant period in this data set.

3.3 Hybrids: AutoPeriod and RobustPeriod

It is much easier to extract a signals' top-k frequencies using the DFT rather than its AC. The DFT contains strong peaks, which can be thresholded. It can, however, include false positives due to spectral leakage. In contrast, the AC contains hills and valleys which require a peak finding algorithm (compare Figure 2 and 3). AC peaks however, when correctly retrieved, capture more accurately the period size estimations [18].

Algorithm 2 Multi-Window-Finder

```
1: procedure MULTIWINDOWFINDER(T, s, e)
2:     D_T ← initialize array with INF of size m
3:     for w ← s to e do
4:         M_w ← MOV_AVG(T,w)
5:         D_T[w] ← SUM(LOG(ABS(M_w − MEAN(M_w))))
6:     end for
7:     M ← FIND_LOCAL_MINIMA(D_T)
8:     top3 = M_{1,3} · [1, 1/2, 1/3]
9:     return MEAN(top3),STD(top3)
10: end procedure
```

AutoPeriod [18] is a hybrid algorithm that combines both approaches to overcome their own limitations. It computes the DFT, thresholds it to retrieve dominant period size candidates, and then validates them using AC. It filters the Fourier frequencies with a 99% confidence interval and assigns the remaining candidates to their closest AC hills or valleys. It then outputs the top location of the selected hills as the dominant periods. From these, the best autocorrelated frequency can be selected as a window size.

Another hybrid approach is RobustPeriod [19], which first removes the trend in a TS, then decouples its periodicities and lastly detects dominate ones using a bespoke filtered DFT and AC. It uses a Hodrick–Prescott (HP) filter to estimate and remove the TS trend, which can negatively impact the AC computation. The procedure then decomposes a TS into multiple frequency ranges (using maximal overlap discrete wavelet transform) to facilitate the detection of multiple dominant periods. Lastly, RobustPeriod detects periods (in dominant frequency ranges) using bespoke modified variants of DFT and AC that use Fisher's significance test for filtering false positives. From these, the most significant frequency is selected as a window size.

3.4 Multi-Window-Finder

Multi-Window-Finder (MWF) [8] is a subsequence-based approach. Its main hypothesis is that the variance in the moving averages is small given a *suitable* window size. This suitable window size then captures a temporal pattern that repeats throughout a TS. In order to determine the width of this pattern, MWF measures the variance for a range of candidate windows and summarizes the best-fitting ones as the window size for the TS.

Definition 6. *The moving average of a TS T, $|T| = n$ and a window size w, is a series of mean values $\mathcal{M}_w := (m_1,...,m_{n-w+1}) \in \mathbb{R}^{n-w+1}$, such that $m_k := \frac{1}{w} \cdot \sum_{j=0}^{w-1} T[k+j]$, for $k = 1,...,n-w+1$.*

Given a suitable window size w, its variance, measured as the distance of the moving average $\mathcal{M}_w$ to its mean $\mu_{\mathcal{M}_w}$, produces a local minimum compared to smaller (or larger) window sizes.

The Multi-Window-Finder (Algorithm 2) takes a TS T, start offset s and end offset e as input. The procedure then calculates the moving averages $\mathcal{M}_w$ over all window sizes between s and e (line 4). While s is set to a low constant value (e.g. 10), e must either be set using domain knowledge or as a fraction of the TS length (e.g. $\frac{n}{5}$). The absolute distances between

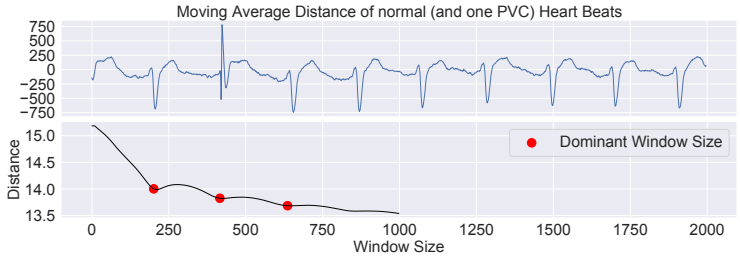

Fig. 4. The moving average distance has distinctive local minima for window size 201 and 416 as well as a more shallow one for 636.

the moving averages $\mathcal{M}_w$ and their means are stored in an array D_T at corresponding offsets (line 5). The local minima in these distances represent suitable window sizes. They are located using a differencing technique (line 7), and the first three are selected and weighted by position (line 8). Thereby, the final window size is represented by the three smallest dominant window sizes. These either represent multiples of temporal patterns in the TS or distinct ones. The algorithm reports the final window size as their average, as well as the associated confidence as their standard deviation (line 9–10).

Its runtime complexity is dominated by the calculation of the moving averages and the distances to their means. A single moving average for a given window size can be efficiently computed in $\mathcal{O}(n)$ differencing cumulative sums of the TS. The absolute distance of the moving average to its mean can then simply be computed in $\mathcal{O}(n)$, too. For a total of $m = e - s$ candidate window sizes, this computation takes $\mathcal{O}(m \cdot n)$. The local minima distances are located in $\mathcal{O}(n)$ and the final window size calculation is computed in constant runtime. Hence, the overall runtime complexity is in $\mathcal{O}(m \cdot n)$. The space complexity is in $\mathcal{O}(n)$, as two additional arrays are used.

Figure 4 shows an example of the moving average distances computed by MWF. The TS (top) shows an ECG recording with normal (and one PVC) heart beats (each one recorded within 250 data points) [9]. The corresponding moving average distances (from 1 to 1k) (Figure 4 bottom) show a substantial local minimum for window size 200 and 416 as well as a more subtle one for 636. MWF uses these three positions to calculate a final window size of 207 which accurately captures one heart beat.

3.5 Summary Statistics Subsequence

Summary Statistics Subsequence (SuSS) [5] is a recent subsequence-based WSS algorithm that compares summary statistics (mean, standard deviation, range of values) computed over subsequences with those computed over the full TS. Its assumption is that these summary statistics of appropriate subsequences are close to those of the entire TS. Hence, their width is a good choice as a window size.

The pseudocode for SuSS is given in Algorithm 3. It takes a TS T as input and calculates its mean, standard deviation and range as a summary statistics vector $stats_T$ of size 3 (line 13). It uses these global statistics to calculate their distance to the rolling statistics of subsequences with changing length. For a candidate window size $w \in [1, \|T\|]$, the rolling summary statistics

Algorithm 3 Summary Statistics Subsequence

```
1:  procedure STATS_DIFF(T, w, stats_T)
2:      stats_w ← (roll_mean(T,w),roll_std(T,w),roll_range(T,w))
3:      stats_diff ← 1/√w · EUCLIDEAN_DISTANCE(stats_T,stats_w)
4:      return mean(stats_diff)
5:  end procedure
6:  procedure SUSS_SCORE(T, w, stats_T)
7:      s_min,s_max ← STATS_DIFF(T,||T||,stats_T), STATS_DIFF(T,1,stats_T)
8:      score ← min-max scale STATS_DIFF(T,w,stats_T) to [s_min,...,s_max]
9:      return 1−score
10: end procedure
11: procedure CALC_SUSS(T, t)
12:     T ← min-max scale T to range [0,...,1]
13:     stats_T ← (mean(T),std(T),1)
14:     lbound,ubound ← EXPONENTIAL_SEARCH(T,stats_T,SUSS_SCORE,t)
15:     w ← BINARY_SEARCH(T,stats_T,SUSS_SCORE,lbound,ubound,t)
16:     return w
17: end procedure
```

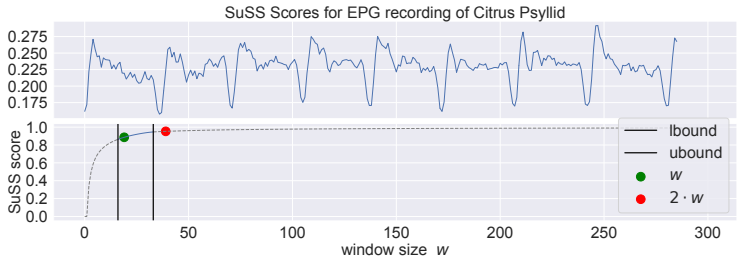

Fig. 5. SuSS locates a narrow range between 16 and 32 (using exponential search) in which the requested window size 20 is found (using binary search), which corresponds to half of the period size.

$stats_w$ are calculated as a matrix such that the i-th row contains the statistics for $T_{i,i+w}$ (line 2). SuSS calculates the Euclidean distances between $stats_T$ and all rows in $stats_w$ and weighs them with the inverse of the root of the window size (line 3). The normalization of the distances corrects a bias for larger windows that are inherently more similar to the full TS. It creates length-invariant distances comparable across window sizes. The mean distance over all window sizes (line 4) is scaled and represents the final SuSS score for w (line 7–9). These scores monotonically increase with w as the statistics of the windows and the TS eventually align. SuSS first conducts an exponential, followed by a binary search to efficiently find the smallest window size w with a score larger than a pre-defined threshold $t \in [0,...,1]$, fixed to a domain-agnostic constant value of 89%. The exponential search identifies a small interval $lbound \leq w \leq ubound$, in which the binary search locates the best w (line 14-16).

The runtime complexity is in $\mathcal{O}(n)$ for a given candidate window size. Searching the candidate space is in $\mathcal{O}(n \cdot \log w)$ for the combined search. As the window size is constraint by n, SuSS has a worst-case complexity of $\mathcal{O}(n \cdot \log n)$. In most applications, however, w is a

Benchmark Name	Type	Number of Data Sets
HEX UCR Anomaly Benchmark 2021 (HUAB) [9]	Anomaly Detection	250
Time Series Segmentation Benchmark (TSSB) [16]	Segmentation	83
ECG heartbeats	Motif Discovery	1
Muscle Activation	Motif Discovery	1

Table 2. Benchmark Data sets used. For motif discovery no annotated benchmark exists.

small constant, which leads to a runtime complexity of $\mathcal{O}(n)$. The space complexity is also in $\mathcal{O}(n)$, as the algorithm uses a matrix of shape $n-w+1\times 3$ to calculate a SuSS score (which is a single real value) for a given subsequence width.

Figure 5 illustrates an EPG recording of an Asian Citrus Psyllid used for studying its feeding behaviour [9]. In the bottom, we show (for illustration) all the SuSS scores between window size 1 to 300. The scores quickly converge as the appropriate window size is independent of the TS length. For a threshold of 89%, SuSS firstly bounds the search space between 16 and 32 and then finds the window size 20 (score 89,2%) which accurately captures half of a period (see the red dot that captures the entire period).

4 Experimental Evaluation

We experimentally evaluate the six WSS methods for anomaly detection, segmentation and motif discovery. We first describe the data sets, methods and evaluation metrics used for the study (see Subsection 4.1). We analyse the performance results for anomaly detection (see Subsection 4.2) and segmentation (see Subsection 4.3) in a quantitative study, and explore the quality of motifs in Subsection 4.4. In order to foster the application and development of WSS algorithms in follow-up works, we provide source codes, Jupyter-Notebooks, visualizations and raw measurement sheets on our website [15].

4.1 Setup

Data Sets: We use the two largest benchmark data sets available for the anomaly detection and segmentation task, and two selected TS from the motif discovery literature (Table 2).

The *HEX UCR Anomaly Benchmark Dataset 2021 (HUAB)* [9] consists of 250 TS from natural, human and animal processes recorded with medical, motion and other sensors. It was released as part of a SIGKDD'21 competition [9] and used to evaluate MWF [8]. Each TS contains an anomaly-free training prefix and a subsequent test suffix, in which exactly one synthetic anomaly is injected.

The *Time Series Segmentation Benchmark (TSSB)* [16] contains 83 TS from a wide variety of device, medical, image, motion and other sensors. It was published in [14] and later extended [5]. Each TS is constructed from one UCR data set by grouping TS by label and concatenating them to create segments with controlled distinctive temporal patterns and statistical properties. The offsets, at which the segments are concatenated, are annotated as CPs.

For motif discovery, we present two use cases. *Muscle Activation* was collected from professional in-line speed skating [12] on a large motor driven treadmill with Electromyography (EMG) data of multiple movements. It consists of 29.899 measurements at $100Hz$

corresponding to $30s$ in total. A muscle activation and recovery takes around 120 ms. *ECG Heartbeats* contains a patient's (ID 71) heartbeat from the LTAF database [13]. It consists of $3k$ measurements at $128Hz$ corresponding to $23s$. The heartbeat rate is around 60 to 80 bpm.

Data Mining Methods: We compare three window-based methods from the anomaly detection and segmentation literature, as well as two algorithms for motif discovery. Note, that we are not mainly interested how these methods compare to each other (per task), but rather how window sizes from the different algorithms influence their performance. We use default parameters for all algorithms.

For anomaly detection, we evaluate discord discovery (with the matrix profile) (MP) [20], isolation forest (IF) and a linear one-class support vector machine (SVM). These methods represent a variety of very different approaches to the problem of anomaly detection, and many new techniques extend their principles. For segmentation, we assess window-based segmentation (with mahalanobis cost) (Window) [17], FLOSS [6] and ClaSP [5]. Window is a baseline approach, and FLOSS and ClaSP are the most recent domain-agnostic methods published. From the motif discovery literature, we chose EMMA [11] and Learning Motifs [7], the two reference implementations for K-Motifs and Latent Motifs.

Evaluation Metrics For the quantitative analysis of TSS and AD, hard and soft metrics exist. The former measures if an algorithm's predictions match the ground truth, while the latter reports the degree to which the predictions precisely match with it. We choose hard metrics suggested in the literature, which provides a fair evaluation in the presence of ties. This is inherently the case for parameter-tuning (in general) and WSS specifically, as all algorithms in Section 3 try to estimate a signal's period. In TSDM, it is common practice to consider a prediction (change point or anomaly) to be correct (TP), if it is in proximity to the ground truth.

F1 score for AD: Assume a TS T of length n, with a train segment from $[1,...,l]$ with $l < n$ and a true anomaly subsequence in the interval $[begin,...,end]$ with $l < begin \leq end \leq n$. Assume an algorithm predicts the timestamp p as an anomaly, we label it correct if it is contained in the true anomaly interval $max(1, begin - 100) \leq p \leq min(end + 100, n)$, considering slack, false otherwise. From this matching, we infer the binary F1 score of either 100% or 0%. We choose a slack of 100 as suggested in [9] for HUAB.

F1 score for TSS: Consider again a TS T of size n and sets of ground truth CPs $cpts_T$ and predicted CPs $cpts_{pred}$, with each location in $[1,...,n]$. The F1 score reports the harmonic mean between precision and recall, where precision reflects the fraction of correctly identified CPs over the number of predicted CPs. Recall computes the number of correctly identified CPs over the number of ground truth CPs. We choose a slack of 1%, as suggested in [5], and allow exactly one correctly predicted CP to count as a TP.

For both TSS and AD, we show critical difference diagrams (as introduced in [3]) to compare the average ranks between approaches per benchmark. The best approaches scoring the lowest (average) ranks over all benchmark TS are shown to the right of the diagram. Groups of approaches that are not significantly different in their ranks are connected by a bar, based on a Nemenyi two-tailed significance test with $\alpha = 0.05$.

Motif discovery is an exploratory task applied to unlabelled data. Thus, it is hard to measure the quantitative effect of WSS. Instead, we did a qualitative analysis with weakly labelled TS, and compared the found motifs with the ones reported in the literature.

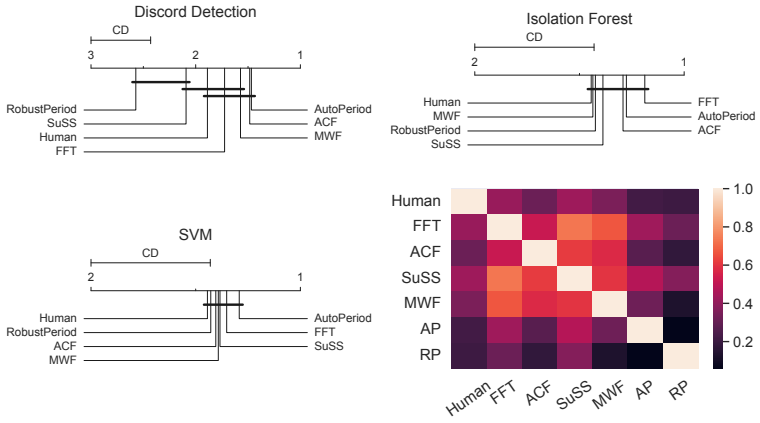

Fig. 6. F1 ranks on 250 benchmark data sets for MP, SVM and IF in combination with human and detected window sizes and window sizes correlation between approaches

4.2 Anomaly Detection

We compare discord detection (with MP), isolation forest (IF) and one-class support vector machine (SVM) in combination with human annotations on the 250 benchmark data sets (HUAB). Figure 6 shows average F1 ranks for the methods.

For discord detection, AutoPeriod ranks first, followed by ACF, MWF, FFT, human, SuSS and RobustPeriod. The first two best-ranking WSS approaches rank significantly better than the last two and the 3rd to 6th approaches rank insignificantly different. The accuracy of the candidates range from 32.8% (RobustPeriod) to 54% (ACF) with high standard deviations between 47-50% (due to the binary scoring). Considering IF (and SVM), the performance differences fade for all WSS algorithms and become statistically indifferent. The mean ranks range minimally, the accuracy is low and between 16.8% and 24.4% (13.2% and 18%) and the standard deviation ranges between 37.5% and 43% (33.9% and 38.5%). Compared to discord detection, MWF and human window sizes perform worse, while FFT and SuSS perform better.

The main finding of these results is that the six WSS methods perform comparably to each other but surprisingly better than human window size annotations. The correlation heatmap in Figure 6 confirms that FFT, ACF, SuSS and MWF have a moderate to strong correlation, indicating all find similar window sizes. AutoPeriod protrudes with weak correlation, but best-ranking results (rank 1 and 2). Hence, AutoPeriod may be concluded as a favourite for WSS in the TSAD task. AutoPeriod is a hybrid of FFT and ACF and finds anomalies in frequency- and time-domain. This might be the reason for its superior performance. However, the differences are not significant. As such, we expected WSS to perform favourably well for anomaly detection, as a change of the period is indicative of an anomaly. The results underline that it is competitive with or even surpassed human annotations.

4.3 Segmentation

Figure 7 contains the average F1 ranks for ClaSP, FLOSS and human annotations for the 83 TSSB benchmark data sets as well as a window sizes correlation heatmap.

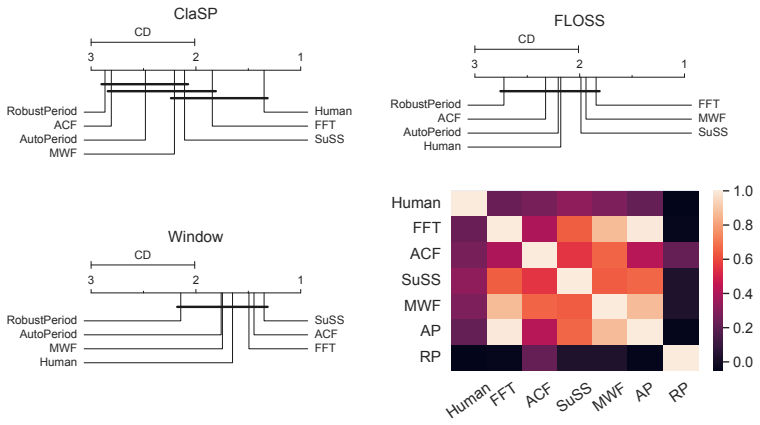

Fig. 7. F1 ranks on 83 benchmark data sets for ClaSP, FLOSS and Window in combination with human and detected window sizes and window sizes correlation between approaches

ClaSP performs best with human windows, followed by FFT, SuSS, MWF, AutoPeriod, ACF and RobustPeriod. While the top-4 approaches do not rank significantly different, only human window sizes perform significantly better than the 3 worst approaches. Performances (standard deviations) range from 69.1% to 88.5% (20.1% to 30.9%). FLOSS (and Window) show only insignificant differences between WSS approaches. The mean ranks are in range of 1.8 to 2.7 (1.4 to 2.1), the mean F1 score is between 56.6% and 62.5% (39.2% and 43.5%) and the standard deviations range in 22.8% to 27.3% (20.7% to 25.3%). In contrast to ClaSP, human annotations rank worse and ACF as well as MWF score better results.

As observed for anomaly detection, the WSS methods perform competitive to human annotations. This is surprising, as the TS have changing periods between segments, such as ”running” and ”walking” in human activity. All methods except RobustPeriod further show moderate to strong relationships (compare Figure 7). FFT and SuSS are both within the top-3 ranking approaches and may be concluded as favourites for this task. Interestingly, the WSS by AutoPeriod is more correlated with the ones from other methods, yet it performs worse (rank 5 and 6). This may indicate that the AutoPeriod detection deteriorates in TS with regime changes, as opposed to FFT and SuSS. However, all differences are not significant.

4.4 Motif Discovery

We present the quality of the found motifs of two challenging real-life data sets for EMMA and LM using human and detected window sizes in Figure 8.

ECG Heartbeats: This data set contains two motifs, a calibration signal with 6 occurrences and the actual heartbeats with 16 occurrences. Two top motif sets as computed by the different methods are shown in Figure 8 (left). The best results could be achieved using EMMA or LM in combination with ACF and SuSS. Both methods found calibration waves as TOP-1 motif and heartbeats as TOP-2 motif (bottom images). The location of the motifs and its lengths are shown in Figure 8 (center). The found periods of the other window selection methods result in unrecognizable noisy signals or no motifs found.

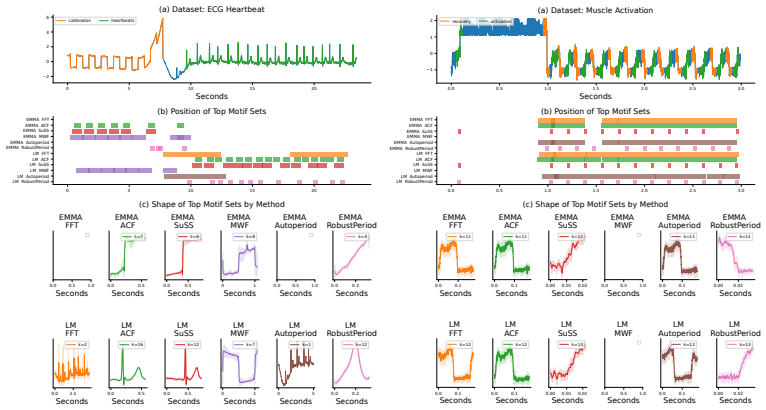

Fig. 8. Found motifs on two popular use cases: ECG heartbeats and inline skating motions

Muscle Activation: The two top motif present in this data set are the activation and the recovery phase of the Gluteus Maximus muscle and have 12–13 occurrences as shown in Figure 8 (right). The best results could be achieved using EMMA and LM using FFT, ACF and AutoPeriod. The TOP-1 motif for EMMA is a whole activation and recovery cycle. The TOP-2 motif found is a shifted variant of the TOP-1 motif and represents - in parts - the recovery phase. MWF failed to find motifs, and SuSS found a too small window size to return meaningful motifs.

Overall, the window lengths of all methods are either too small or too large to distinguish between recovery and activation. ACF, however, seems to find period lengths that produce meaningful motifs on both data sets, the other competitors did not succeed to give good lengths on all data sets. The problem of TSMD is different from that of the other two use cases mentioned before, as the TS are non-periodic. In motif discovery, we search for frequent similar substructures. This substructure may or may not be correlated to a period of the TS. WSS, however, tries to identify a period in the TS. Thus, by design it fails for TSMD. On our supporting website [15], we show the results of two additional (non-periodic) data sets for motif discovery.

5 Summary

We applied window size selection (WSS) to three distinct time series data mining tasks. Segmentation and anomaly detection require the input data to be periodic. As such, WSS even surpasses human annotations. In anomaly detection, a change in the period is even a sign of an anomaly. Segmentation is further characterized by changing window sizes between segments. In motif discovery, the TS itself must not be periodical, but we search for frequently appearing, similar subsequences. As such, we did not see a clear winner for all scenarios covered. From the WSS methods, the window sizes from FFT, ACF, SuSS and MWF are correlated and do not lead to significantly different performances. AutoPeriod, however, ranked highest for anomaly detection and FFT and SuSS are the two best-ranking whole and subsequence-based methods for the segmentation task and have low runtimes. In

a global ranking (across both tasks and all methods), FFT ranks first, AutoPeriod and SuSS second, ACF and MWF third, followed by human window sizes, and RobustPeriod. For motif discovery, the presented approaches gave unsatisfactory results, as the size of non-trivial substructures in TS cannot easily be determined by its period size. Hence, more research is needed in data mining domains where the TS is non-periodic or contains changes in its period length. Also, incremental and online detection of changing window sizes is an interesting research area. Future work could further evaluate one of the mentioned TSDM tasks in more detail or explore more of its algorithms, benchmarks and evaluation metrics.

References

1. Blázquez-García, A., Conde, A., Mori, U., Lozano, J.A.: A review on outlier/anomaly detection in time series data. ACM Computing Surveys (CSUR) **54** (2021)
2. Dau, H.A., Bagnall, A.J., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C., Keogh, E.J.: The ucr time series archive. IEEE/CAA Journal of Automatica Sinica **6** (2019)
3. Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. The Journal of Machine Learning Research **7** (2006)
4. Elfeky, M.G., Aref, W.G., Elmagarmid, A.K.: Periodicity detection in time series databases. IEEE Transactions on Knowledge and Data Engineering **17** (2005)
5. Ermshaus, A., Schäfer, P., Leser, U.: ClaSP - Parameter-free Time Series Segmentation. arXiv (2022)
6. Gharghabi, S., Yeh, C.C.M., Ding, Y., Ding, W., Hibbing, P.R., LaMunion, S.R., Kaplan, A., Crouter, S.E., Keogh, E.J.: Domain agnostic online semantic segmentation for multi-dimensional time series. DMKD **33** (2018)
7. Grabocka, J., Schilling, N., Schmidt-Thieme, L.: Latent time-series motifs. TKDD **11**(1) (2016)
8. Imani, S., Keogh, E.: Multi-window-finder: Domain agnostic window size for time series data. MileTS (2021)
9. Keogh, E., Dutta Roy, T., Naik, U. and Agrawal, A: Multi-dataset time-series anomaly detection competition. https://compete.hexagon-ml.com/practice/competition/39/ (2021)
10. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing sax: a novel symbolic representation of time series. DMKD **15**(2) (2007)
11. Lonardi, J., Patel, P.: Finding motifs in time series. In: Workshop on Temporal Data Mining (2002)
12. Mörchen, F., Ultsch, A.: Efficient mining of understandable patterns from multivariate interval time series. DMKD **15**(2) (2007)
13. Petrutiu, S., Sahakian, A.V., Swiryn, S.: Abrupt changes in fibrillatory wave characteristics at the termination of paroxysmal atrial fibrillation in humans. Europace **9**(7) (2007)
14. Schäfer, P., Ermshaus, A., Leser, U.: Clasp - time series segmentation. CIKM (2021)
15. Supporting Material: https://github.com/ermshaua/window-size-selection (2022)
16. Time Series Segmentation Benchmark: https://github.com/ermshaua/time-series-segmentation-benchmark (2021)
17. Truong, C., Oudre, L., Vayatis, N.: Selective review of offline change point detection methods. Signal Processing (2019)
18. Vlachos, M., Yu, P.S., Castelli, V.: On periodicity detection and structural periodic similarity. In: SDM (2005)
19. Wen, Q., He, K., Sun, L., Zhang, Y., Ke, M., min Xu, H.: Robustperiod: Robust time-frequency mining for multiple periodicity detection. SIGMOD/PODS (2021)
20. Yeh, C.C.M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H.A., Silva, D.F., Mueen, A.A., Keogh, E.J.: Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. ICDM (2016)