IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 22 June 2023, accepted 12 July 2023, date of publication 17 July 2023, date of current version 24 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3295995

RESEARCH ARTICLE

Variable Length Motif Discovery in Time Series Data

M. VAN ONSEM[1,2], V. LEDOUX[2], W. MÉLANGE[2], D. DREESEN[2], AND S. VAN HOECKE[1]
[1]IDLab, Ghent University–imec, 9052 Ghent, Belgium
[2]Skyline Communications, 8870 Izegem, Belgium

Corresponding author: M. Van Onsem (matthias.van.onsem@skyline.be)

This work was supported by Flanders Innovation and Entrepreneurship (VLAIO) and Skyline Communications through the Baekeland Project under Grant HBC.2019.2588.

ABSTRACT The detection of recurring behavioral patterns in time series data, also called motif discovery, is a crucial step for mining insights in complex time series data, especially in complex environments where manual monitoring is not feasible. However, current state-of-the-art algorithms fall short in their applicability in production environments (due to static motif length, lots of user defined parameters, only providing the best motif pair, etc.). In this paper, a variable length motif discovery method is proposed based on the Matrix Profile which focuses on industrial applicability. It works in noisy and periodic environments, returns only unique motifs (meaning same shape motifs are grouped together as one) and only requires one distance matrix calculation. The method was benchmarked on synthetic data as well as publicly available real world key performance indicator (KPI) data from telecom providers and shows adequate accuracy in finding both short and long motifs in the same time series.

INDEX TERMS Motif discovery, network monitoring, time series, matrix profile.

I. INTRODUCTION

The operation of companies, governments and other institutions are largely influenced by the proper workings of devices, such as servers, network nodes, machines, etc., as a result of an increasing investment in automation. As the number of these devices, as well as the complexity of how they interact with each other, is getting beyond human understanding, demand for automated monitoring tools has risen accordingly over the last decades. These automated tools track a number of *key performance indicators* (KPIs) such as CPU-load, bitrate and signal strength of each device through time to generate a digital overview of the whole operation [1]. This digital overview provides the user with important insights, such as abnormal behavior and predictions that a service is reaching its limits, to reduce the amount of information that the user has to process manually and allows for a more targeted way of solving problems. Additionally, automated root cause analysis can be used to further drill down on anomalies (i.e., abnormal behavior) triggered on multiple KPIs by providing the user with a starting point to start his investigation from. As the user gets pointed directly to the root cause of problems on the moment or even before they start happening, downtime is reduced significantly, with the benefits increasing even more for more complex setups.

As KPIs are tracked through time as time series (i.e., a series of values obtained at successive times), certain patterns emerge that characterize certain events such as rain fades or backup cycles. These events are often recurring and hold valuable information when searching for anomalies (e.g., backup cycles happen every night at 3am, or power surges happen at sundown when the lights turn on). The process of identifying these recurring patterns, called *motif discovery*, plays a crucial role in identifying abnormal behavior, as well as offers a tool for extracting other insights such as relations between parameters: when a backup spike is identified every time a network switch outputs a lot of traffic, one can assume the switch is connected to the backup server.

In a typical production environment there are many different types of KPIs that are monitored. Some KPIs tend to produce noisy time series with big motifs, others

The associate editor coordinating the review of this manuscript and approving it for publication was Kostas Kolomvatsos.

have very clean behavior with smaller motifs. To enable motif discovery in production environments, the following requirements should therefore be met:

1) Motifs of all temporal sizes should be detectable: Motifs of 2 hours long as well as motifs of 5 minutes long should be detectable by the same method without the need of manual interaction of the user, as illustrated in Figure 1. There are two reasons why this requirement is crucial. First, polling time and motif length can differ greatly from KPI to KPI (e.g., a transcoding session can take 30 minutes while power outages may result in motifs of 5 minutes). As it is unfeasable for a user to configure a hyperparameter, i.e., motif length, for every KPI, motifs of all sizes should be detected automatically. Second, motifs can also differ in size for the same KPI (i.e., transcoding sessions and backup spikes can be visible on the same CPU, with large variety in temporal size). If only one motif size is detected, it reduces the usability of the discovery tool drastically in most situations [2], [3].
2) Motifs of all resolutions should be detectable: Production environments contain data with and without noise. The amount of noise should not impact the accuracy of the motif discovery method. Hence, patterns with a lot of noise should be detectable with the same accuracy as patterns with little noise [4].
3) All of the occurrences of the same motif should be grouped together under the same discovered motif label. This ensures that when two different motifs are discovered with multiple occurrences each, automation scripts can be attached to each motif, allowing more granular control.
4) Motifs should not contain multiple repetitions. When a motif occurs two times in a row, it is expected two occurrences of a single motif are identified. The algorithm should for example return a daily pattern instead of a concatenation of 5 times that same daily pattern. This ensures matching accuracy and speed. Motifs can be used for detecting anomalous behaviour for example, meaning that if a pattern that should occur daily does not occur, it should trigger an anomaly as fast as possible, if the motif includes multiple repetitions of the daily pattern that suddenly does not occur, it might take several days to conclude it did not occur or not detect it at all depending on the anomaly size.

Because of these requirements, a motif should be as large as possible without being seasonal. This ensures user visibility (having one daily pattern instead of 24 1 hour patterns makes it easier for a human to read) while ensuring detection speed and accuracy (Requirement 4).

As will be shown in the Section III, current state-of-the-art solutions fail to check all of these requirements and are therefore only applicable in a limited way or not at all. In this paper, a novel motif discovery approach is proposed that can extract motifs of different sizes by looking for matching subsequent motif candidates. The method filters out duplicate motifs so only unique motifs are found, and it can also detect seasonal behavior to ensure one-period motifs are identified. The method does not rely on brute-forcing all motif sizes, only needing one distance matrix calculation. This paper also introduces a distance metric that takes into account the scale of the motif when grouping them together, reducing the amount of false positives in the process.

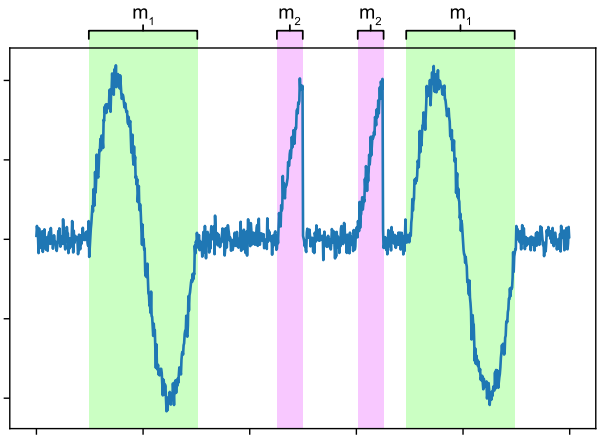

FIGURE 1. An illustration of a time series with different sizes. As described in Requirement 1, both large and small motifs should be detectable without manual interaction.

The remainder of this paper is structured as follows: Section II will give a short overview on the contributions of this paper and its scope. Section III discusses the current state-of-the-art methods with their strengths and shortcomings. Section IV explains the proposed approach in detail. Section V will show some benchmarks on both synthetic and real world KPI data. Section VI will discuss the limits of the algorithm and how they could be solved in future work. Lastly, Section VII will formulate a conclusion.

II. CONTRIBUTIONS AND SCOPE

This paper proposes a novel way of discovering exact motifs in time series using a pairwise distance matrix and a matrix profile. It aims to increase applicability in industrial application by reducing computation time, the amount of hyperparameters and postprocessing needed to extract valuable information such as the occurrences of each motif. The contributions to the current state of the art are as follows:

1) A novel way to detect motifs of all sizes with only one distance matrix calculation, reducing computation time compared to comparable methods. The method is around 3-8 times faster to comparable methods.
2) A scale aware distance metric that reduces noise in the pairwise distance matrix. The distance metric is normalized so that thresholds can be set independently of the time series scale on the y-axis.
3) The proposed method only requires two hyperparameters, mainly a distance threshold and window size. No minimum or maximum motif size or top k

parameter has to be given in advance, reducing the amount of manual work in the deployment of the method.

4) The algorithm returns a set of different motifs grouped with all their occurrences. This makes the result easier to interprete and attach to other automated services without additional steps (e.g. alerting if a certain motif does or does not occur).
5) The performed benchmarks also show an increase in accuracy, detecting motifs that are not detected with VALMOD.

III. RELATED WORK

When extracting motifs, scrolling windows of a certain size w are typically used to extract all possible subsequences from a given time series [4], [5]. A straightforward way of finding *nearest neighbour motifs* would be to calculate all pairwise distances between subsequences using z-normalized Euclidean distance. This generates a *distance matrix* [5], a symmetric matrix where each value represents the distance between the subsequences starting at the row and column index. Using this matrix, the lowest value from each column can be extracted to generate a *matrix profile*, where an area with a width of one window size centered at the main diagonal is typically excluded to prevent *self matching*. As this matrix profile represents the likelihood of a subsequence being a motif (a lower distance value means a higher similarity between two subsequences), nearest neighbour motifs can be extracted by finding the lowest distance values with their respective index pairs [5]. So called *K-frequent motifs* can also be extracted using a threshold value on the distance matrix and extracting the number of values below this threshold in each column. Selecting the highest counts results in the best candidates for the most frequent motifs [4].

One of the main downsides of these techniques however is that the motif size has to be known in advance (requirement 1) [3]. To solve this, brute force methods have been proposed that calculate distance matrices based on different window sizes and combine each matrix profile to extract the biggest motif when ranges overlap [3]. As this requires a lot of computational resources, more lightweight methods like VALMOD have been proposed which estimate whether a larger pattern needs to be calculated [2]. This allows to quickly get the best nearest neighbour motif on every window size in a certain predetermined range. A postprocessing step is needed however to remove duplicate motifs and merge overlapping ranges (requirement 3). Seasonal behavior will not be detected by this technique (requirement 4), meaning the extracted motif can consist of a concatenation of the same pattern multiple times. Additionally, the running time of VALMOD can still pose a problem when scaling up to production environments in amount of parameters.

As these approaches tend to be very sensitive to temporal scale differences between patterns, approaches like *dynamic time warp* (DTW) have been proposed, which finds a warping path between two sequences [6], [7]. However, this technique tends to be slower than a lockstep distance metric, as it needs to calculate more point to point distances for each sequence pair [8].

FIGURE 2. A visual representation of the matching accuracy of SAX using the normal distribution. Subsequences with exactly the same SAX representation are hard to find, and allowing some mistakes generate shifted matches that only match by coincidence.

To improve performance, symbolic approaches have been suggested such as *symbolic aggregate approximation* (SAX) [9], [10], [11], [12]. This technique uses *piecewise aggregate approximation* (PAA) to aggregate the time series by dividing it into the concatenation of subsequences of a predefined size w and summarizing each part by its average. Each average value is then assigned a symbol by normalizing the aggregated subsequence and defining threshold values on the normal distribution curve so that the area under the curve between each subsequent threshold point is the same. Each aggregated sequence value then gets a symbol assigned to it unique to the two threshold values it falls under in the normal distribution curve. This reduces the number of possible values in a sequence to a predefined number, making it easier to find similar patterns. One of the main problems of SAX is its matching accuracy due to the way it extracts its thresholds. Figure 2 and Figure 3 illustrate this matching accuracy problem by looking for matches for the queried area. As thresholds are defined by the normal distribution curve, these thresholds do not change from case to case and result in a lot of *close calls* points that are near these threshold points. Extracting the distribution from the time series itself and using area under the curve to define the threshold points improves the accuracy in some cases, but only to a certain degree. Due to the above mentioned challenges and the fact that grouping similar motifs together is vital in production environments, SAX tends to be a challenging method to work with.

Machine learning approximation techniques like autoencoders have also been proposed to reduce computation time [13], [14]. These approaches consist of training an

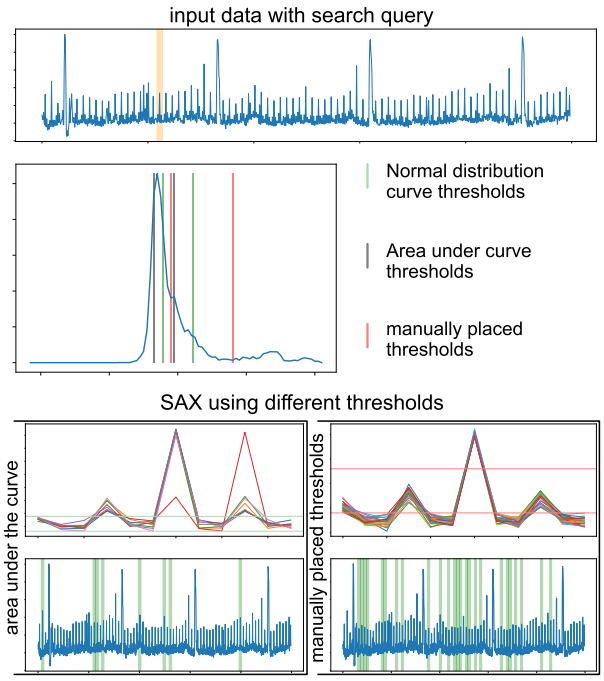

FIGURE 3. A visual representation of SAX using the value distribution extracted from the time series via a scrolling window. Taking the area under the curve generates a better matching result than normal sax with fewer false positives. These results can be improved further with smarter threshold extraction methods, as manual placement of these thresholds eliminates the false positives altogether. The matching results are still too inconsistent for production deployment however.

autoencoder onto preprocessed subsequences and clustering the latent space vectors to identify recurring behavior. As the compressed latent space is used for motif discovery however, the discovery process is not exact and suffers from similar matching issues like SAX. The same holds for with methods such as wavelet transforms, but they are not commonly used in motif discovery environments [4].

In conclusion, current state-of-the-art methods do not meet all of the requirements and are therefore not suitable for deployment in industrial applications. A summary of the current best candidates is illustrated in Table 1. Starting with the matrix profile [5], which is not able to fulfill any of the requirements. Pan matrix profile [3] does slightly better as it is able to find a maximum size of a motif but does so at a huge computational cost. Variable resolution motifs are discoverable on the condition that they can be seen in the starting window size. VALMOD improves upon this method by reducing the computational cost drastically, but does not improve on the other requirements in any significant way. SAX methods can be used to detect motifs of multiple sizes [12], but seem to show a lot of false positives and false negatives in matching subsequences, which is crucial for finding large motifs.

IV. EVENT DRIVEN MOTIF DISCOVERY

As current state-of-the-art methods largely fail to extract variable length motifs without the need of post processing steps or within viable time to make them applicable in industrial applications, this paper proposes a novel approach of motif discovery which identifies unique variable length motifs and is able to find the correct motif in repeating signals and only requires one distance matrix calculation to work. The steps of the algorithm are as follows. First, a distance matrix is calculated using a custom distance metric, as described in Section IV-A and Section IV-C. The matrix values are then screened for subsequent low values to extract large motifs, which is described in Section IV-D. Lastly, Section IV-E and Section IV-F will explain some of the post processing that is done such as grouping all motif occurrences together and detecting seasonality.

TABLE 1. Comparison between current state of the art methods for motif discovery with respect to the requirements defined in Section I.

	Variable length	Variable resolution	Occurrence grouping	Seasonal data
The matrix profile [5]	- -	- -	- -	- -
Pan matrix profile [3]	-	-	- -	- -
VALMOD [15]	+	-	- -	- -
SAX [12]	-	-	- -	- -

A. PROPERTIES OF A DISTANCE MATRIX

Given a time series of length n and window size w, a pairwise distance matrix is created: an $(n-w) \times (n-w)$ matrix where the coefficient on position $[i, j]$ equals the distance $D(S_i, S_j)$ between the two subsequences T_i, T_j of size w starting at index i and j of time series T. As $D(S_i, S_j)$ is usually symmetric (the distance between T_i and T_j tends to be the same as the distance between T_j and T_i), this matrix tends to mirror around the main diagonal, meaning only half the matrix needs to be calculated to have all pairwise distance values. The main diagonal itself is composed of zeroes, as this line represents the distances between every subsequence to itself. Usually, an exclusion zone of $\frac{1}{2}w$ on each side is defined around the main diagonal to prevent so called *self matches* or *trivial matches* [16]. Each diagonal stacked on top of the main diagonal represents the time series being matched with the shifted version itself, e.g., as illustrated in Figure 4, the diagonal stacked 6 rows above the main diagonal contains information on matching the time series with itself shifted 6 to the right.

B. DEFINITIONS

The following terminology is used:

- **Distance function**: a function used to compute the distance values between two subsequences.
- **Event**: a distance value in the pairwise distance matrix that is beneath a certain threshold. The corresponding subsequence pair of length w is considered similar or a *match* and can be considered occurrences of the same motif or occurrences of part of a motif.

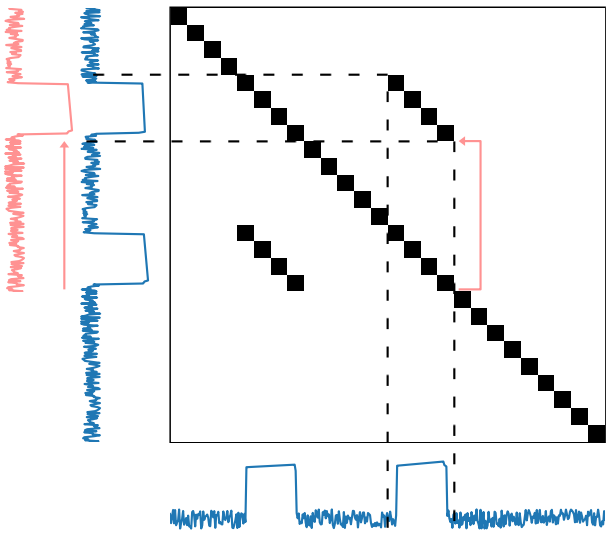

FIGURE 4. A time series being matched to itself in a pairwise distance matrix. Black areas depict matching subsequences. The distance to the main diagonal depicts how much the time series is shifted over itself, following along the diagonal gives an idea of the lengths of the parts of the time series that match closely to their shifted counterparts.

C. DISTANCE MATRIX CALCULATION

Using a time series T and user defined window size w, a pairwise distance matrix is computed. Typically, a z-normalized Euclidean distance function D_{z-norm} is used [5], as defined in equation 1, where T denotes the time series in which motifs are searched for and i and j denote the start positions of the subsequences S checked with size w. μ_i and μ_j describe the average of the subsequences whereas σ_i and σ_j describe their standard deviations. This distance metric removes the scale differences of two subsequences, only calculating the differences in shape. Using this function brings the benefit that every distance value has an upper boundary of $2\sqrt{w}$, which makes it easier to define a generic threshold.

$$D_{z-norm}(S_i, S_j) = \sqrt{\sum_{l=0}^{w-1} \left(\frac{T_{i+l} - \mu_i}{\sigma_i} - \frac{T_{j+l} - \mu_j}{\sigma_j} \right)^2} \quad (1)$$

However, network monitoring tools often operate in noisy environments where this function tends to produce noisy distance matrix results, as shown in Figure 5. As scale differences tend to have an impact on the expected motif groups, a more scale-aware distance $D_{scale-aware}$ function is preferred. It is important that the scale difference is defined in a generic way however, meaning the global scale should not impact the detection accuracy of matching subsequences (e.g. the same time series scaled up or down should produce the same scale differences). To achieve this, this paper proposes a scale distance D_{scale} which consists of the relative difference of the standard deviations of both subsequences, as described in equation 2.

$$D_{scale}(S_1, S_2) = \frac{max(std(S_1), std(S_2))}{min(std(S_1), std(S_2))} - 1 \quad (2)$$

To preserve the upper boundary of the distance function, a cap value c is applied to the scale distance, which results in a capped scale distance $D_{scale-capped}$, as illustrated in equation 3. In other words, when c is set to 2, all subsequences that are two or more times as prominent will result in a distance value of 1.

$$D_{scale-capped}(S_1, S_2) = \frac{min(D_{scale}, c)}{c} \quad (3)$$

To get the final distance value, a vector is formed with both the capped scale distance and the normalized z-normalized Euclidean distance. The size of this vector, which has an upper bound of $\sqrt{2}$ is then normalized and taken as the scale-aware distance $D_{scale-aware}$, as described in equation 4.

$$D_{scale-aware}(S_1, S_2) = \frac{\sqrt{\left(\frac{D_{znorm}(S_1,S_2)}{2\cdot\sqrt{w}}\right)^2 + D_{scale-capped}(S_1, S_2)^2}}{\sqrt{2}} \quad (4)$$

A comparison of both the z-normalized Euclidean distance and scale-aware distance is visualized in Figure 5. As can be seen, now a clear distinction between prominent and not so prominent patterns is visible without reducing the matching accuracy on smaller spikes.

D. MOTIF EXTRACTION

After calculating the pairwise distance matrix using the $D_{scale-aware}$ distance function, a predefined threshold is applied to generate a binary matrix M_{bin}, applying equation 5, where v represents each value in $D_{scale-aware}$ and t the predefined threshold.

$$M_{bin(i,j)} = \begin{cases} 1, & \text{if } D_{scale-aware(S_i,S_j)} \leq t \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

A 1-value in this matrix is called as an *event*. Note that only the upper half of the matrix needs to be calculated, as it is symmetric. A matrix profile MP using the scale-aware distance matrix is also calculated.

Using M_{bin}, every diagonal is then ranked by counting all the events on each diagonal and ordering them from high to low. This ensures the diagonals with the most or largest motifs are processed first. The process is illustrated in Figure 6.

All diagonals are processed using the extracted order to look for motifs. In broad terms, this is done by searching for events present on that diagonal. When an event is encountered, a motif *candidate* is extracted by enlarging it using the matrix profile. The motif candidate is then validated by calculating the distance of the enlarged window size and testing it to the event threshold. The details of this process are as follows.

Finding events on the diagonal being processed is a trivial task. The diagonal is scanned in the binary matrix until an index $[i, j]$ is encountered where the coefficient is 1, meaning the corresponding time series subsequences of size w starting at i and j look alike. It is possible that the subsequences of say size $w + 1$ also look alike, i.e. the actual motif size

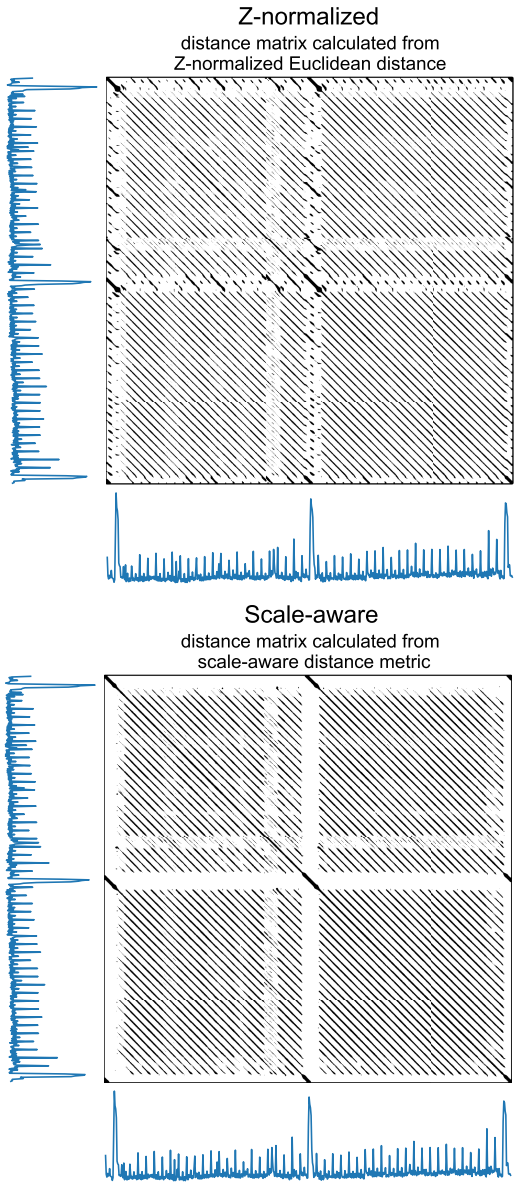

FIGURE 5. A comparison between the z-normalized and the scale-aware distance metric (black regions depict a match by applying a matching threshold on the matrix values). As the large spike sometimes matches with smaller spikes, it makes the detection of seasonality and motif occurrences difficult. This noise is not present when using the scale-aware distance metric.

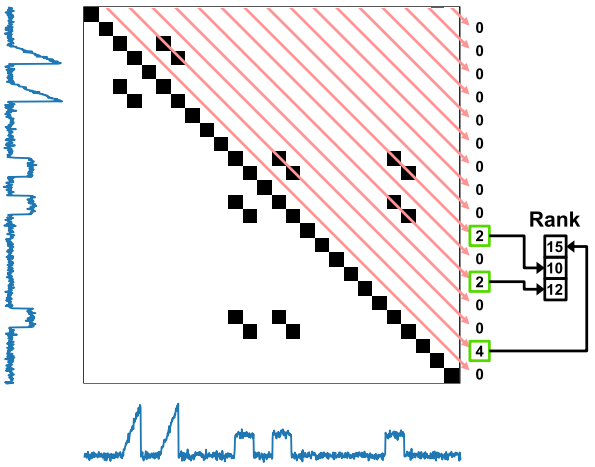

FIGURE 6. Diagonal ranking is done by counting every event on each diagonal and ranking them from high to low. As the 15th diagonal contains four events, it is ranked first, followed by the 12th and 10th diagonal respectively, as they both count two events.

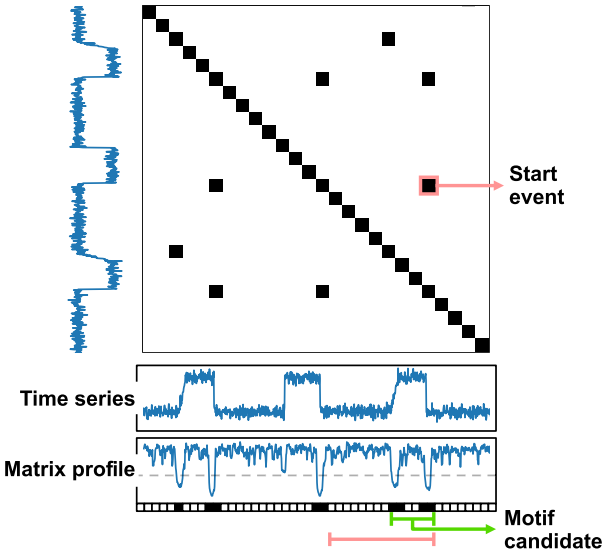

FIGURE 7. A visualization of the motif discovery process. When an event is detected on a diagonal, the matrix profile is checked for events to enlarge the motif up to a motif size equal to the distance to the main diagonal (as that is the maximum size a motif on that diagonal can have without overlapping, this is marked in red). All the events in that window on the matrix profile define the range of the motif candidate.

is not known at this point. To find the actual motif size, note that it can not exceed a fixed maximum, namely the distance between the diagonal being processed and the main diagonal (which depicts the amount of shift that is applied to the time series before matching). Taking a range that exceeds the shift would mean the subsequences at that cross section would overlap each other. The matrix profile values around the starting event in a range equal to the *maximum motif size - w* are checked against the event threshold (The matrix profile value is chosen to enlarge the motive to prevent close call events while enlarging the motif). If they are below the event threshold, the value can be considered part of the motif. The process is illustrated in Figure 7. Noteworthy is that gaps between events part of the same motif are allowed. This means noisy parts in a recurring segment do not cause the motif to be broken up in small parts when detected.

The motif candidate will consist of a list of event indices that potentially belong to the same motif. From this list, a range can be defined from the first point in the indices list to the last summed by w. This size is capped to the shift value to prevent overlap. Say for example the 5th diagonal contains [[11, 6], [13, 8], [14, 9]] as a motif candidate with $w = 10$, the motif candidate range will be $[11 - 15]$ and $[6 - 10]$ respectively, as $14-(11-1)+10 > 5$ and is therefore capped.

Algorithm 1 Hampel Filter Algorithm

S = Matrix profile values of the initially extracted motif range
$\sigma = 3$
k = constant scale factor for normal distribution (≈ 1.4826)
$Motifs$ = empty list
X = All non-zero values in S
$X_{median} = \text{median}(X)$
$X_{distance} = k \cdot \text{median}(\mid X - X_{median} \mid)$
$S_{selected}$ = all indices of S where $S - X_{median} < \sigma \cdot X_{distance}$
return $S_{selected}$

Exceeding the cap would result in ranges $[11 - 24]$ and $[6 - 19]$, resulting in overlapping subsequences. These candidate ranges are then used to calculate the distance value between the two subsequences on the cross section in the distance matrix. In the example, this would be $D([11 - 15]$ and $[6, 10])$. This distance is then tested against the event threshold. If the test succeeds (i.e., the subsequences have a distance below the event threshold and are therefore considered an event or motif), the motif candidate is considered one motif, which is taken to the next step.

If the test fails (i.e., the subsequences have a distance above the event threshold and are therefore considered not an event nor motif), the motif candidate can still hold so called *submotifs*, meaning certain parts in the extracted range are motifs on their own.

To detect these submotifs, the *distance to diagonal* number of matrix profile values are checked for outliers using a modified Hample filter, as described in Algorithm 1. This ensures smaller drops in the matrix profile in that range, which depict submotifs, are kept. Using the value indices, submotifs are extracted by grouping subsequent indices. For example, when putting a range of 10 values in the Hampel filter returns indices $[1, 2, 3, 6, 8, 9]$, the submotifs are $[1, 2, 3]$, $[6]$, $[8, 9]$ respectively. Lastly, a verification step is done by calculating the distance between the submotif ranges for every submotif to check if they are indeed motifs on their own. If the check passes, the submotif is considered a motif.

E. MATCHING MOTIFS

As motifs regularly occur more than twice in a given time series, the method looks for all motif occurrences and filters out all motifs detected in the previous step. To do so, first, the distance profiles are calculated of the found motif. As the motif has been defined as two subsequences matching with each other, the distance profiles of both subsequences should be calculated: i.e., for each of the two subsequences, the distance to every other subsequence (of the same length as the extracted motif) in the time series is calculated. These distance profiles describe at what index in the time series a subsequence matches with the found motif. In a perfect world, the distance profiles of these two subsequences are exactly the same, but this is rarely the case in production environments. Therefore, these values are converted into binary values by testing them against the event threshold. The results represent the starting positions of the subsequences in the time series that match. They are then merged together by checking for overlapping ranges and keeping one of the two results when the overlap exceeds a user defined threshold $t_{overlap}$. The method is described in Algorithm 2.

Algorithm 2 Finding Motif Occurrences

R_{m1}, R_{m2} = Motif ranges, meaning the ranges of both subsequences that matched
D_{m1}, D_{m2} = Distance profile indices of both R_{m1} and R_{m2} ranges respectively that are below t
$motif_{size}$ = motif size
i_{list1} = empty list
$i_{templist}$ = empty list
for each i = index in D_{m1} **do**
  if $i - 1$ in $i_{templist}$ or empty($i_{templist}$) **then**
    append i to $i_{templist}$
  else
    append $i_{templist}$ index x where $D_{m1}[x]$ is lowest to i_{list1}
    clear $i_{templist}$
  end if
end for
append $i_{templist}$ index x where $D_{m1}[x]$ is lowest to i_{list1}
clear $i_{templist}$
i_{list2} = empty list
for each i = index in D_{m2} **do**
  if $i - 1$ in $i_{templist}$ or empty($i_{templist}$) **then**
    append i to $i_{templist}$
  else
    append $i_{templist}$ index x where $D_{m2}[x]$ is lowest to i_{list2}
    clear $i_{templist}$
  end if
end for
append $i_{templist}$ index x where $D_{m2}[x]$ is lowest to i_{list2}
clear $i_{templist}$
for each i_1 = index in i_{list1} **do**
  for each i_2 = index in i_{list2} **do**
    if $\frac{|i_1 - i_2|}{motif_{size}} < t_{overlap}$ **then**
      remove i_2 from i_{list2}
    end if
  end for
end for
i_{list} = merge(i_{list1},i_{list2})
return i_{list}

After finding all of the motif occurrences, the ranges are excluded from the binary matrix by setting both the row and column of the motif matches to 0. This prevents

Algorithm 3 Motif Discovery Broad Steps

```
T = input time series
w = user defined window size
t = user defined threshold
O = empty motif output list
calculate Distance matrix M and matrix profile S calculated with T and w using D_scale−aware
calculate M_bin using M and t
sort diagonals of M_bin by number of events
for each diagonal d in diagonals do
    for each event e in d do
        enlarge motif using e and S
        if motif is not valid then
            check for submotifs
        end if
        match found motif(s) on rest of the time series
        append motifs and their matches to O (unique motifs and all its occurrences are one item)
        exclude motifs and matches from M_bin
    end for
end for
```

occurrences of the same motif to be found again, as well as noisy submotifs in the same range. The exclusion process is illustrated in Figure 8.

This process is repeated until all diagonals are processed. The high level steps of the whole process are described in Algorithm 3

F. MOTIF POST PROCESSING

As the distance matrix diagonals are processed in the order of most events, some unwanted behaviour can occur. More specifically, when a time series shows a periodic signal (e.g., a sine wave), there is a chance of finding a motif which includes 2 or more sine waves instead of one. To solve this issue, an auto correlation function (ACF) [17] is calculated on a subsequence or occurrence of each discovered motif. If the ACF dips below 0.5 and then increases to a high value again (≥ 0.95), it indicates repeating behavior in the subsequence. When this occurs, the motif size is corrected to the location of the first high value it can find in the ACF. If the ACF never dips below 0.5, the motif data is detrended and screened again, as globally high ACF values are often present in time series with trends.

Lastly, the motifs are ordered based on prominence, where non prominent motifs are filtered out as false positives. To do this, the average standard deviation of every motif occurrence is calculated per motif, resulting in a standard deviation value for every unique motif. These values are then ranked from high to low, meaning the most prominent motifs are put first. In order to filter out false positives, the global prominence of the time series is calculated for each motif by moving over the time series with a scrolling window and extracting the standard deviation at each step. The size of the scrolling

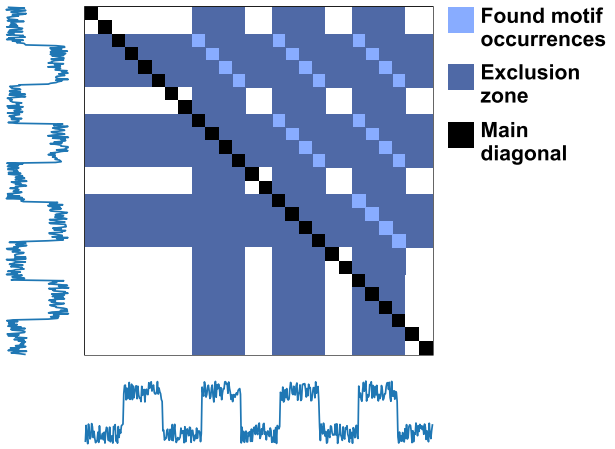

FIGURE 8. Visual representation of the exclusions of the motif matches from the matrix. All of the matching range combinations are excluded from the matrix, as well as their respective columns and rows to combat noise.

window is equal to the size of the motif being processed. From all the standard deviations, the quartile values as well as the median value is extracted. If the prominence value of the motif is lower than the median value of the time series increased with 1.5 times the interquartile range (a value chosen experimentally), the motif is considered not prominent enough and discarded. The resulting method as well as all used datasets are made publicly available on github [18].

G. COMPLEXITY

Comparing the computational complexity is difficult, as state-of-the-art algorithms differ in scope. VALMOD calculates the best motif pair for each subsequence size, but does not filter out duplicates or picks the best motif candidates overall [2]. Pan matrix profile does pick the best motif size candidate, but again does not filter out duplicates [3].

In worst case, the computational complexity of the proposed algorithm in this paper is $O(n^2)$ with n being the time series size. This is comparable to VALMOD (which has a worst case complexity of $O(n^2 \cdot R)$ where R being relative to the smallest and largest motif size). Pan matrix profile also has a complexity of $O(n^2 \cdot R)$. As the maximum motif size is also tied to the time series length (the maximum motif size could be $\frac{n}{2}$), this brings the complexity of the Pan matrix profile to $O(n^3)$.

The prototype code [18] has a space complexity of $O(n^2)$, as a binary matrix is kept in memory to find the motifs. Because the distance matrix in the proposed algorithm is only used to rank diagonals based on number of events however, a similar version could be implemented by just using the matrix profile together with an index vector of the best matching pair, which would reduce space complexity to $O(n)$. This does increase the number of calculations though, as for each found motif, the matrix profile values of the matched indexes have to be recalculated and will not always output

the same motifs, as the startpoint of the discovery process is changed.

V. BENCHMARKS

To benchmark the proposed method, three datasets are used:

1) Synthetic dataset
2) Numenta NY taxi dataset [17]
3) Skyline's in-house dataset

The benchmark consists of running the proposed method with the same hyperparameters on each time series and verifying the output manually. In total, there are three user-defined hyperparameters. The first one is the window size of the subsequences used to calculate the distance matrix. This is set at 20 points in the shown figures, a value which was empirically chosen. The threshold, which decides whether a distance value is a match, is set to 0.2 and the size difference cap is set to 4 (the relative size difference between two subsequences that results in the maximum size difference in the scale-aware distance measure, as explained in Section IV-C). These values were chosen experimentally. The results are verified manually as the datasets used are not labeled for motifs. The Skyline dataset also does not include any y-axis unit labels. This should not impact results, as motifs by definition are not unit specific. All results are publicly available for verification.

Compute time of the algorithm is also compared to VALMOD. The Series Distance Matrix framework implementation in Python was used to calculate VALMOD [15]. As both codebases are Python based, this ensures an accurate comparison between the two methods. For VALMOD, a detection range between 10-300 points is set. This boundary is chosen based on the motifs present in the dataset. The computation times can be found in Table 2. The method proposed in this paper is about 3 to 8 times faster than VALMOD depending on the chosen window size, as only one distance matrix calculation is needed to extract motifs. It is worth noting that the proposed method does not need a motif range defined and can detect larger motifs out of the box if there are any present in the time series. The computation time tends to reduce with higher window sizes. There are two key reasons for this. First, computing a distance matrix with larger window sizes tends to be lower than smaller ones [16]. Second, by increasing the window size, smaller motives become invisible because it gets blurred out in the distance result by the added non motif values. This generally reduces computation time for processing the matrix. There is a notable exception in the Numenta dataset at 50 points window size however. One of the causes is the amount of noise present in this case, as shown in Figure 9. At small window sizes, this generates a lot of gaps in the distance matrix, reducing the amount of events that have to be processed by the motif discovery method. As the size increases, these gaps get bridged because the noise is not as prominent to the subsequence anymore, generating more events and therefore increasing computation time, this is illustrated in Figure 10. If the window size increases even more, the needed computation time reduces again because of the distance matrix calculation needing less time.

TABLE 2. Computation time to find motifs in seconds (lower is better). Time series are approximately between 1000 to 10000 points in size. The proposed method was tested with window size 10, 20, 50, 100 and 200 respectively.

	Synthetic dataset	Numenta NY taxi dataset	Skyline dataset
VALMOD (10-300)	100	320	5319
Prop. method (10)	10	33	2108
Prop. method (20)	7	33	1234
Prop. method (50)	7	36	826
Prop. method (100)	6	29	796
Prop. method (200)	5	24	785

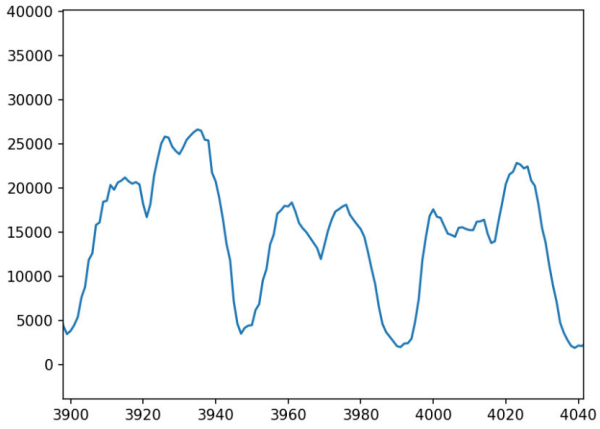

FIGURE 9. A small sample of the Numenta taxi dataset. The top of each motif daily pattern has some noise of 10-30 point in size.

A. SYNTHETIC DATASET

The synthetic dataset consists of a collection of different patterns to test out the discovery capabilities. The tested time series are illustrated in Figure 11. The data contains a noisy signal and generates some motifs of different shapes and sizes. As shown, all of the motifs are extracted regardless of their size, periodicities and frequencies. All of the motif occurrences are also grouped together, meaning no duplicate motifs are identified as unique.

Figure 11 shows the method is capable of identifying the unique motifs with all their occurrences of motifs of multiple sizes, periodicity and frequency, which is a big factor in the applicability in production contexts.

VALMOD largely finds the same motifs in the best motif pairs, albeit at other window sizes, as shown in Figure 12. One notable difference is how the scaled distance metric is capable of distinguishing high and low spikes from each other. Another difference is the fact that VALMOD needs additional processing to group motifs together, as it only returns the best matching subsequence pairs out of the box. This is shown in the example, as the first time the sine wave is detected is at a window size of 271 points. It is also

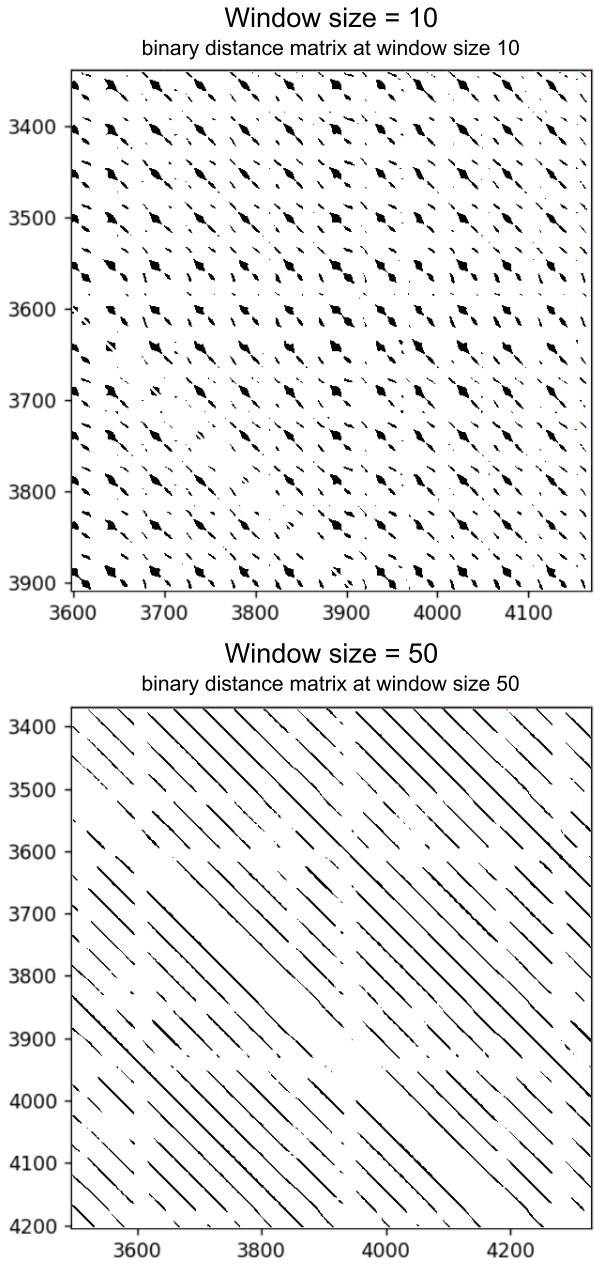

FIGURE 10. A sample of the binary distance matrix at window size 10 and 50, with black representing an pair match. Window sizes that are smaller than the noise present in the time series tend to produce gaps in the distance matrix. As the window size increases, these gaps are bridged, generating more events on that diagonal.

not capable to detect any periodicity in the found motifs, which means it can generate results where multiple pattern occurrences are grouped together as one motif, which is noticable in how the sine wave is detected.

B. NUMENTA NY TAXI DATASET

The Numenta dataset [19] is a labeled anomaly detection benchmark for time series data. It consists of multiple categories, such as cloud monitoring, traffic data, etc. The dataset used in this benchmark is the New York city taxi

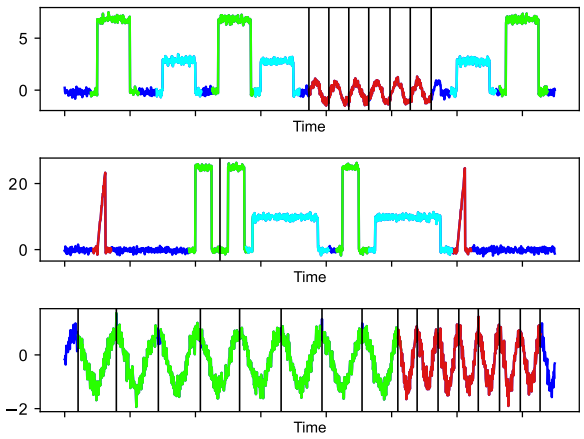

FIGURE 11. The results of the proposed method on the synthetic benchmark. With the found motifs color coded with same color patterns depicting occurrences of the same motif which are grouped together by the method. For repetitive motifs, division lines are displayed to depict what motif is detected. As shown, all of the motifs are detected and grouped correctly.

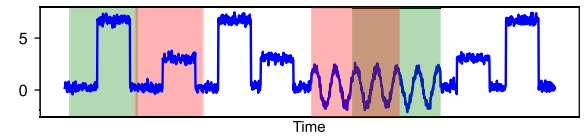

FIGURE 12. The results of VALMOD on the synthetic benchmark with two distinct pairs selected (with green matching to red marked regions). As illustrated, the same motifs are detected as the proposed method, but don't make a distinction between differences in scale and does not detect multiple repetitions in a motif.

passenger count during the day over a period of a few weeks. The signal is periodic and consists of a daily pattern, as illustrated in Figure 13. The method identifies the weekly and daily pattern and consistently matches on their occurrences. The last two regions on which no daily and weekly occurrences are matched also coincide with the anomaly labels provided on the Numenta benchmark, meaning the mismatches are expected behavior. While there is some dispute on the accuracy of the Numenta ground truth labels, the regions where no motif occurrence was found also coincides with discords found using the matrix profile method [20].

As the signal is periodic, VALMOD finds the expected motif. The proposed method extends the capabilities of VALMOD by not growing the motif size beyond its seasonality of a week or a day. Added to that, the proposed method also finds all of the motif occurrences and groups them as such, making it easy for a user or automation script to process the found motifs further.

C. SKYLINE DATASET

The Skyline dataset consists of 25 time series from production environments from customers of the network management vendor Skyline Communications. The data has been picked

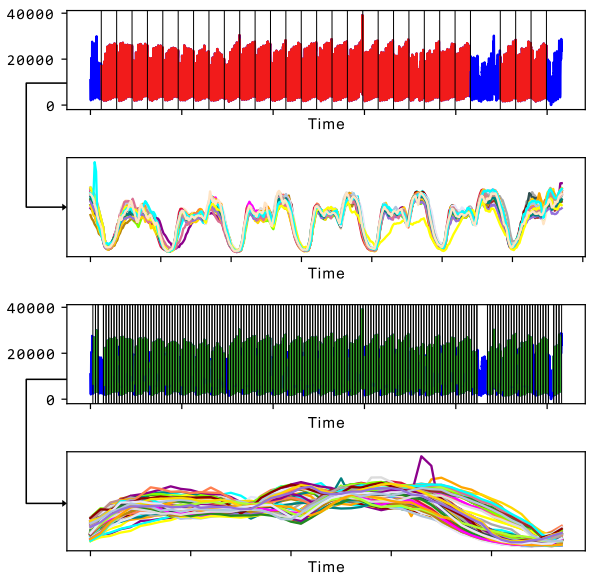

FIGURE 13. The results of the Numenta benchmark. The extracted motifs are shown on the time series with division lines, and displayed in detail below it. As is shown, A weekly pattern is detected with 7 daily bumps. This motif is matched on the whole time series except the regions where anomalies are present. A daily pattern is also extracted, matching consistently on the time series except on anomalous days.

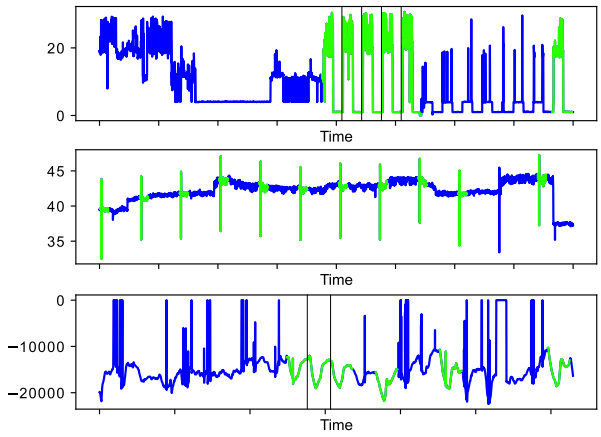

FIGURE 14. A sample result of the Skyline benchmark. Division lines are displayed when two occurrences of the same pattern follow directly up on each other.

manually on the basis of the presence of a motif. A sample of the dataset is illustrated in Figure 14. The full dataset is included with the source code for verification.

As telecom providers usually monitor millions of parameters in real time on mainstream server hardware, time series data gets aggregated over time to ensure system performance. This means that the length of time series in telecom production environments is rather limited to a few thousand data points.

As seen in the sample results, all prominent motifs are found and are matched on their occurrences. It is also worth to note that noise does not tend to ever match on found patterns

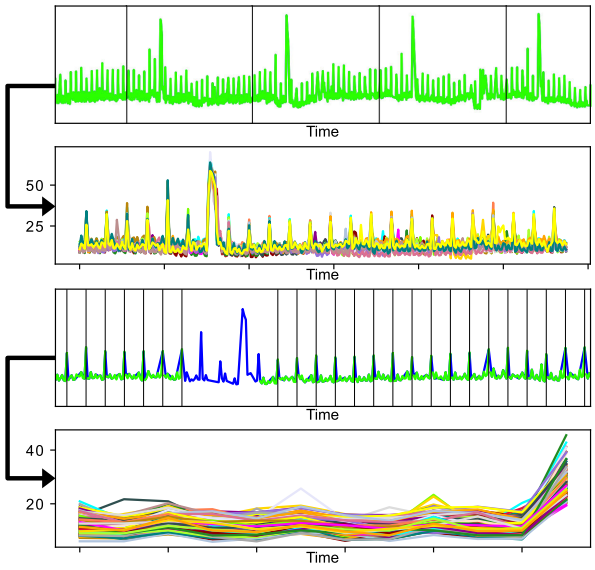

FIGURE 15. Another result of the Skyline dataset in detail. 2 motifs are present here with very different sizes, yet they are both detected and grouped successfully with one distance matrix calculation. Additionally, motifs that are smaller than the predefined window size are still detectable, as shown in the second motif found consisting of 12 points, with a window size of 20 points.

due to the scale-aware distance measure, adding an extra layer of robustness.

The variable length property of the algorithm can also be illustrated in this real world use case, as is illustrated in Figure 15. Two motifs are present here, differing widely in size (one spanning 288 points or 1 day, whereas the other is merely 12 points or 1 hour). As shown, the proposed method is able to extract both motifs with only one distance matrix calculation. Added to that is the fact that the small motif is smaller than the set window size of 20 points, but was still detectable at the right size because of the distance to the main diagonal.

When running VALMOD on this case, only the larger spike is detected as a motif. At window size 10, the larger spike is more consistently matched than a smaller spike. As VALMOD only returns the best matching pair, it tends to miss motifs in cases where multiple different motifs are present of the same size, as one pair is preferred over the other. This is illustrated in Figure 16, where VALMOD only manages to find the level shift down, missing the smaller level shifts upwards.

D. SUMMARY

An overview of how the method compares to current state of the art methods with respect to the requirements defined in Section I is shown in Table 3. As there is no motif size range that has to be known in advance, it improves upon VALMOD with respect to finding variable length motifs. It does have the same weaknesses when it comes to finding motifs at variable resolutions however. This is not an issue when the defining features of a motif are visible in the window size, as gaps

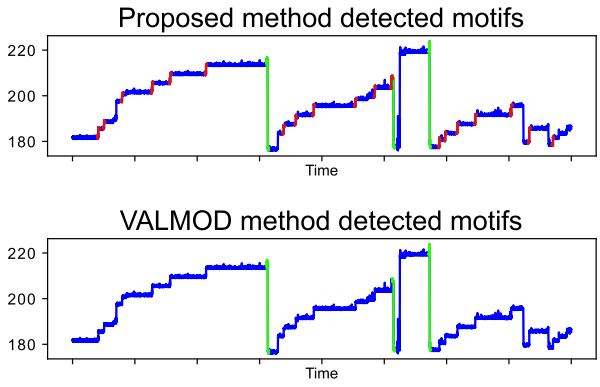

FIGURE 16. Comparison between results generated by the proposed method compared to the ones generated from VALMOD. The motifs marked in red are never found with VALMOD, as they are never a best matching pair. All the motifs found by VALMOD (i.e. subsequences that were part of a best matching pair) in the 10-300 window size range are marked in green. Motifs marked in the same color are grouped together by the proposed method.

TABLE 3. Comparison between the current state of the art methods to the proposed method with respect to the requirements defined in Section I.

	Variable length	Variable resolution	Occurrence grouping	Seasonal data
The matrix profile [5]	- -	- -	- -	- -
Pan matrix profile [3]	-	-	- -	- -
VALMOD [15]	+	-	- -	- -
SAX [12]	-	-	- -	- -
Proposed method	+ +	-	+ +	+

allow for noisy parts of a motif to be ignored, as shown in the synthetic benchmarks. One aspect the proposed method improves drastically on is identifying and grouping all motif occurrences of a motif, which comparable methods do not do. It also shows periodic patterns are detected at the correct size, which again improves current state of the art methods.

VI. LIMITATIONS AND FUTURE WORK

One limitation of the proposed algorithm is memory usage. As the binary matrix is used to detect motifs, it is kept in memory, resulting in a n^2 space complexity. While the impact is limited in the benchmarked use cases with an average 8000 points per time series, it limits the method from discovering motifs in very large time series without averaging. As the binary matrix is only used as a starting point to motif candidate selection however, there are at least two possible solutions that can be investigated to reduce the n^2 space complextity problem. First, all distance matrix values can be calculated as we need them, discarding them afterwards to reduce memory usage. This would remove the binary matrix from memory and only requires the already discovered motif ranges to be kept in memory to exclude them from being discovered as the next motif candidate. There is a computational cost to this however, as the binary matrix will have to be calculated twice (once to calculate the matrix profile and rank the diagonals, and once to discover the starting positions of the motif candidates). Compared to VALMOD, the added computation time would not have an impact on the benchmark conclusions, as it can only double the computation time in worst case. Another possible solution is to remove the binary matrix in the discovery process altogether in favor of the calculated matrix profile. Instead of fetching the motif candidates from the binary matrix, the lowest matrix profile value is taken as a start until the lowest matrix profile value crosses the event threshold. This approach would require to keep the *matrix profile index* [5] in memory to identify the motif candidate pair and recalculate the matrix profile every time a motif is found so found motifs are excluded from future calculations. Apart from an added computational cost, this solution would also change the found motifs however, as there is no diagonal sorting that prioritizes which motifs are found first. Experiments will need to be performed to check if the result is desirable.

Finding motifs across multiple resolutions is also not directly addressed by this method (requirement 2). This is due to the fact that motifs are still discovered from events calculated from a single window size. It is worth noting that if some motif characteristics are defined within the chosen window size this does not pose an issue, as gaps between events of the motif candidate are allowed (e.g. if a motif is very noisy but consists of a level shift up and a level shift down, the whole motif will be detectable by those level shifts). A possible solution to this problem could be to average the time series using a wavelet transform or piecewise aggregate approximation and reapply the motif discovery method on the result. Further research is needed to determine how results from different aggregation levels should be merged to get the best result however.

VII. CONCLUSION

In this paper, a novel motif discovery method was proposed with the focus on production applicability. As shown in the benchmarks, the method is capable of identifying motifs of multiple sizes (fulfilling requirement 1) in significantly less time than the comparable method VALMOD. It is also capable of detecting periodicity in motifs and identifying only unique motifs and grouping those with all its occurrences (fulfilling requirement 3 and 4).

Future work is needed to find motifs across multiple resolutions (requirement 2). While allowing gaps allows for some resistance to noise, it is limited to the motif features being visible in the set window size. A possible solution would be to compute the Piecewise Aggregate Approximation (PAA) of the time series with multiple bucket sizes and finding motifs on each PAA level. This would increase the amount of computations, but only to limited degree, as the number of distance matrix points decreases with each PAA bucket size increase.

DATA AVAILABILITY STATEMENT

The datasets generated during and/or analyzed during the current study are available in Skyline Communications

Github repository, https://github.com/VOMatthias/Variabl-Length-Motif-Discovery-Data-And-Code

REFERENCES

[1] *Software-Defined Networking (SDN) for Media Explained*, Skyline Commun., Izegem, Belgium, 2021.
[2] M. Linardi, Y. Zhu, T. Palpanas, and E. Keogh, "Matrix profile X: VALMOD—Scalable discovery of variable-length motifs in data series," in *Proc. Int. Conf. Manage. Data*, May 2018, pp. 1053–1066.
[3] F. Madrid, S. Imani, R. Mercer, Z. Zimmerman, N. Shakibay, and E. Keogh, "Matrix profile XX: Finding and visualizing time series motifs of all lengths using the matrix profile," in *Proc. IEEE Int. Conf. Big Knowl. (ICBK)*, Nov. 2019, pp. 175–182.
[4] S. Torkamani and V. Lohweg, "Survey on time series motif discovery," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 7, no. 2, p. e1199, Mar. 2017.
[5] C. C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 1317–1322.
[6] K. Mehrotra, C. Mohan, and H. HuaMing, *Anomaly Detection Principles and Algorithms*. Berlin, Germany: Springer, 2017.
[7] S. Alaee, K. Kamgar, and E. Keogh, "Matrix profile XXII: Exact discovery of time series motifs under DTW," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2020, pp. 900–905.
[8] O. Gold and M. Sharir, "Dynamic time warping and geometric edit distance: Breaking the quadratic barrier," *ACM Trans. Algorithms (TALG)*, vol. 14, no. 4, pp. 1–17, 2018.
[9] E. Keogh, J. Lin, and A. Fu, "HOT SAX: Efficiently finding the most unusual time series subsequence," in *Proc. 5th IEEE Int. Conf. Data Mining (ICDM)*, Mar. 2005, pp. 8–15.
[10] J. Lin, E. Keogh, and S. Lonardi, "Experiencing SAX: A novel symbolic representation of time series," *Data Mining Knowl. Discovery*, vol. 15, pp. 107–144, Oct. 2007.
[11] B. Chiu, E. Keogh, and S. Lonardi, "Probabilistic discovery of time series motifs," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2003, pp. 493–498.
[12] Y. Gao and J. Lin, "Efficient discovery of variable-length time series motifs with large length range in million scale time series," 2018, *arXiv:1802.04883*.
[13] F. K.-D. Noering, Y. Schroeder, K. Jonas, and F. Klawonn, "Pattern discovery in time series using autoencoder in comparison to nonlearning approaches," *Integr. Comput.-Aided Eng.*, vol. 28, no. 3, pp. 237–256, Jun. 2021.
[14] K. Bascol, R. Emonet, E. Fromont, and J.-M. Odobez, "Unsupervised interpretable pattern discovery in time series using autoencoders," in *Structural, Syntactic, and Statistical Pattern Recognition*. Merida, Mexico: Springer, Dec. 2016, pp. 427–438.
[15] D. De Paepe, S. V. Hautte, B. Steenwinckel, F. De Turck, F. Ongenae, O. Janssens, and S. Van Hoecke, "A generalized matrix profile framework with support for contextual series analysis," *Eng. Appl. Artif. Intell.*, vol. 90, Apr. 2020, Art. no. 103487.
[16] Y. Zhu, Z. Zimmerman, N. S. Senobari, C.-C. M. Yeh, G. Funning, A. Mueen, P. Brisk, and E. Keogh, "Matrix profile II: Exploiting a novel algorithm and GPUs to break the one hundred million barrier for time series motifs and joins," in *Proc. 16th Int. Conf. Data Mining (ICDM)*, 2016, pp. 739–748.
[17] T. Hill and P. Lewicki, *Statistics: Methods and Applications: A Comprehensive Reference for Science, Industry, and Data Mining*. Tulsa, OK, USA: StatSoft, 2005.
[18] M. Van Onsem. (2023). *Variable Length Motif Discovery Data and Code*. [Online]. Available: https://github.com/VOMatthias/Variable-Length-Motif-Discovery-Data-And-Code
[19] A. Lavin and S. Ahmad, "The numenta anomaly benchmark (white paper)," Numenta, Redwood City, CA, USA, Tech. Rep., 2017, doi: 10.1109/ICMLA.2015.141.
[20] R. Wu and E. J. Keogh, "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 3, pp. 2421–2429, Mar. 2023.

M. VAN ONSEM received the M.S. degree in electronics and ICT engineering from Ghent University, where he is currently pursuing the Ph.D. degree in cooperation.

He has been with Skyline Communications, Izegem, Belgium, since 2019, as a Machine Learning Research Engineer. His research is mainly focused on anomaly detection and insights mining on time series data.

V. LEDOUX received the M.S. and Ph.D. degrees in computer science from Ghent University.

In 2007, she was a Postdoctoral Fellow with Ghent University. Since 2014, she has been with Skyline Communications, as a Machine Learning Research Engineer, where her research focuses on time series insights mining. Her research was in the field of numerical analysis.

W. MÉLANGE received the M.S. and Ph.D. degrees in computer science engineering from Ghent University.

He is currently with Skyline Communications, as a Machine Learning Research Engineer, where he focuses on big data insights mining. His research mainly focused on queuing.

D. DREESEN received the M.S. and Ph.D. degrees in math from KU Leuven.

He is currently with Skyline Communications, as a Machine Learning Research Engineer, where he focuses on big data insights mining. His research mainly focused on product manifolds and compression.

S. VAN HOECKE received the degree in computer science from the Engineering Department, Ghent University, in 2003, and the Ph.D. degree in computer science engineering and efficient service management in healthcare from the Department of Information Technology, Ghent University. She is an Associate Professor of semantic intelligence with the IDLab, Ghent University–imec. Her research focuses on combining machine learning and semantic technologies for predictive maintenance and predictive healthcare. She has published more than 150 publications in her field.