IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 November 2024, accepted 8 December 2024, date of publication 19 December 2024, date of current version 3 January 2025.

Digital Object Identifier 10.1109/ACCESS.2024.3520337

RESEARCH ARTICLE

Feedback-Driven Pattern Matching 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, imec, Ghent University, 9052 Ghent, Belgium
[2]Skyline Communications, 8870 Izegem, Belgium

Corresponding author: M. Van Onsem (matthias.vanonsem@ugent.be)

This work was supported in part by the Baekeland Project funded by VLAIO and Skyline Communications under Grant HBC.2019.2588.

ABSTRACT While motif discovery methods have come a long way over the years, they generally match occurrences based on the similar shape of the whole subsequence. As patterns in production network monitoring environments, which monitors and manages entire infrastructures of millions of device metrics over time, frequently exhibit more complex characteristics such as differences in temporal size or expected noise, these methods often remain insufficient for accurately tracking important behavioral patterns such as backup cycles or transcode sessions. This paper therefore proposes a feedback framework that allows a user to select additional motif ranges to be included or excluded from the model. The method uses a distance matrix of subsequences to extract common patterns from feedback samples and defines temporal rules on how these patterns are allowed to occur. The technique was tested on synthetic data as well as production network monitoring data and a publicly available human motion primitives dataset. The tests show that the recall score can be significantly improved with the proposed feedback system, increasing from 37% to 95% while also maintaining a perfect precision score. This is achieved by providing only one to three feedback samples as input. While the scope of this paper is limited to shape based features, the proposed technique can also be used for less exact patterns such as changepoints in noise.

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

I. INTRODUCTION

As the number of active devices in corporations and other organisations has skyrocketed over the years, so has the interest in monitoring Key Performance Indicators (KPIs) to ensure their stability. These KPIs are monitored over time as time series (a sequence of values that are typically equally temporally spaced) to allow a clear overview of the KPI behavior. This behavior can then be used to extract key insights such as detect anomalies or discover discords and motifs, which in turn allow the user to quickly follow up on maintenance tasks or interventions when the system does not behave as intended.

As the number of KPIs easily reaches into the millions in typical production network monitoring environments, the interest in extracting these insights automatically has grown significantly. Motif discovery, which identifies recurring patterns (i.e. subsequences in a time series), is a key insight that allows improvement of automated anomaly detection, alerting the user on significant events [1]. For instance, video processing servers typically exhibit recurring block like patterns when a video input is being transcoded (i.e. being converted from one format to another for compatibility in streaming over ip). Detecting this behavior can ensure that system reboots do not occur during activity or that booking schedules are followed.

Automatic motif discovery techniques typically assume the general shape of the whole subsequence is equally important to finding all of its occurrences however [2]. This can introduce mismatches in some cases where certain noise can occur that is irrelevant to the matching decision. For example, a CPU typically exhibit a block pattern when a transcode is in process. However, during the transcode, several factors can impact the shape drastically, such as other processes

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

running in parallel introducing spikes. These should not have an effect on the detection of the transcode session, which only focuses on the block wave increase and decrease being present. When considering each datapoint in the subsequence equally important in finding occurrences, this can lead to unreliable alarming or logging tools, which in turn tend to be ignored. Additionally, certain motifs can exhibit differences in their length over multiple occurrences. The time needed for a backup to finish for example is influenced by the amount of data and the load on the system, which can vary at each cycle. To achieve reliable monitoring, these temporal differences should be taken into account when matching.

Finally, while a fully automated discovery tool which is capable of finding these more complex motif types without human intervention is desirable, they should not introduce false positives by so called coincidental matches to the system. This is crucial in production network monitoring environments where there is typically a high amount of KPIs. Therefore, a hybrid approach where high confidence motif types are automatically discovered without human intervention combined with a feedback framework could provide a potential solution which combines the best of both worlds. This minimizes coincidental motif discoveries while also providing an approach that can be deployed effectively in generic environments where more complex behavior needs to be tracked.

This paper therefore introduces a novel way of refining motif definitions with user input and finding all occurrences based on these refinements. The technique builds further upon classic motif discovery methods based on distance matrix calculations [3], [4], [5], [6], [7]. The main contribution consists of a method that allows the user to select false negative and false positive ranges from an initial motif discovery result and then automatically extracts relevant identifiers from these. These identifiers are then used to provide the user with an improved matching result. It improves the current state of the art in the following ways:

1) User input: Current state of the art methods match discovered motifs based on the similarity of the total subsequence shape, which does not always produce the desired result, as will be shown in Section IV. The proposed method offers a simple way for users to refine the motif definition by defining additional occurrence ranges in the time series.
2) Minimal input: The proposed method does not require an extensive training set of true positives and true negatives in order to work. With only two samples, a model can already identify relevant features and adjust the matching accordingly. Benchmarking shows one to three samples can achieve a recall increase from 37% to 95% while maintaining a 100% precision.
3) Variable length motif definitions: Occurrences of a single motif are often static in size. In many cases however, recurring behavior is not always equally long in duration. An example of this is visualized in Figure 1, where a backup produces a consistent shape based pattern in the free memory KPI. The duration of the backup is tied to how large the files are that need to be backed up. The proposed algorithm allows a user to mark differently sized backup patterns as a true positive, after which the system learns what temporal range a backup typically has. Benchmarking shows the method has no problem matching motif occurrences where the temporal size ratio is over 4:1.
4) Feedback framework for time series: The proposed method, while being tested on exact comparisons in this paper, could also be used for non shape based metrics, such as noise factors or trends [8].
5) White box: The model can explain what key features were extracted and inform the user why an occurrence was detected as the same pattern. This gives the user more control and a clear overview on how the monitoring system works.

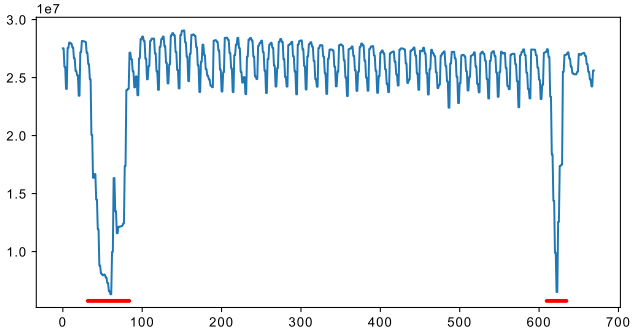

FIGURE 1. An example of a motif generated by a backup process in memory. The backup creates consistent spike patterns, marked in red, but differs in length based on the backup size.

This paper mainly focuses on improving motif discovery recall scores in production network monitoring environments through feedback. The proposed technique can however be deployed to find similarity between verified samples of the same class in any sequence based problem where distances can be calculated between elements. Other use cases can include fault prediction by analyzing historic data between verified incidents for similar behavior. This in turn can give an alert when these behavioral similarities occur again, therefore offering a proactive maintenance method. The technique can also be used in activity tracking, where similarity between a few demonstrations can be used to find future occurrences of the monitored activity with fewer errors.

II. RELATED WORK

Over the years, a lot of algorithms have been developed with the goal of finding consistent repeating patterns in time series data. Exact distance matrix based methods such as the pan matrix profile [4] and VALMOD [3] are capable of extracting variable length motifs with few manual hyperparameters. These methods have been extended over time with improvements such as noise reduction techniques to

improve performance of the z-normalized Euclidean distance metric, which tends to be very noise sensitive [9]. Exact distance matrix methods have therefore become quite capable on discovering motifs without human intervention. One limitation however is that occurrences of a discovered motif are always identified using a distance profile. This means the decision on which characteristics in the time series are important tends to be solely influenced by the deviation on the y-axis compared to the rest of the subsequence. While this might work for a wide number of cases, some patterns tend to have noise in certain regions that is expected or differences in temporal length between occurrences. Exact distance matrix methods typically penalizes these characteristics, making the system unreliable to track these behavioral patterns. Additionally, these techniques also generally do not offer any feedback options to refine motif definitions to improve matching in such cases, reducing their flexibility in environments with many different KPI types.

Several non-distance matrix discovery methods have also been proposed such as Symbolic Aggregate Approximation (SAX) based methods [10], [11], which translate an input time series into a sequence of discrete symbols. As their main goal is to reduce computational resources, they generally struggle with the same problem as distance matrix based methods, even more so as lighter fluctuations are often aggregated into the same symbol. Moreover, as SAX tends to have a higher chance of mismatching occurrences when subsequence values are outside of a normal distribution [6], the need for the possibility to give feedback only increases.

Dynamic Time Warp (DTW) [12] discovery methods have been introduced to overcome temporal differences between motif occurrences. The method does this by finding the lowest cost warping path that matches all values of the input subsequence, returning the cumulative cost as a distance value. While the method performs well in cases where the overall shape is the same, it still penalizes any noise present and processes every input value with the same weight. This is not desired in noisy environments were certain parts of a motif should be ignored for a perfect match.

Neural network approaches use autoencoders to detect motifs. Input sequences are compressed in a latent space which is then clustered together [13], [14]. As the latent pattern space can be retrained, these approaches do offer a way to provide feedback by transforming the latent vector to be removed from the cluster range, but doing so requires a lot of true positives and true negative samples. It would require too much human effort to provide these labels consistently in production environments.

As motif discovery approaches generally do not support user feedback out of the box, classification methods can be used on top of the motif discovery result as a motif matching model. This approach would train a generic classification technique on the initial motif discovery results, and retrain the model with user feedback to refine the matching. Several classification methods have been proposed over the years. Simple methods include nearest neighbour (NN) methods, which can be applied on top of distance measures such as DTW [15]. More complex methods such as deep neural network based classifiers have been proposed as a flexible algorithm that can be deployed in a wide variety of fields, including time series [16], [17]. They calculate the probability that an input sample, which in this case is a motif occurrence, belongs to a certain class by transforming them through multiple layers of connected neurons. While raw input values can be used, image based convolutional neural networks have also been deployed in time series classification by transforming input sequences to a more dense format such as Gramian Angular Fields [18]. Support Vector Machines (SVM) solve binary classification problems by computing the best hyperplane that maximizes the division between classes an have also been applied in time series classification [19], [20]. Decision trees, often combined in ensemble methods, have also been used in time series for classification problems [21], [22], [23].

One major downside of these classification algorithms however is the amount of samples needed to provide accurate results in a time series use case where a lot of features are present per sample [17], [24]. Because of the large amount of KPIs typically monitored in production network monitoring environments, the number of labeled expert samples is very low, making these generic classification methods impractical.

In conclusion, while a lot of progress has been made in the field of motif discovery over the years, they typically focus on one consistent block that exhibits shape based similarity. They are therefore less suited in use cases where complex motif definitions are present such as differences in length and irrelevant noise that should not impact the matching outcome. While generic classification methods can provide a solution for this, the high amount of features per motif sample and scarcity of feedback samples typically given in a production network monitoring environment makes them infeasible. Therefore, the need remains for an effective way of modeling complex motif definitions from very limited feedback samples.

III. FEEDBACK DRIVEN PATTERN MATCHING

The method proposed in this paper incorporates user input to modify the way a pattern is matched in a time series. As input, one or more verified patterns are submitted (i.e., subsequences that are confirmed to be part of the same motif class). These are subsequences with equally temporally spaced data points that are part of the same pattern the user wants to find occurrences of. They can be defined completely via human labeling, or partially by initial motif discovery method outputs. Using these verified patterns, a model is constructed with the requirement that it needs to match on all verified patterns. It does so by identifying similarities and modelling temporal rules between them. As the user only has to select additional ranges for the method to process, human effort is very minimal.

The proposed algorithm consists of three building blocks. First, a distance matrix is used to extract similarity between

two verified patterns. Then, a warping path is defined on similar parts that aims to maximize a cumulative similarity score between the two verified patterns. Third, using this path, temporal rules between the extracted similar parts are defined for matching.

This section is structured as follows. First, the distance matrix structure needed for pattern refinement is discussed in Section III-A. From this structure, Section III-B explains the method which extracts similarity features which will be used as the defining characteristics for the motif class. Section III-C discusses two approaches on scalability beyond two patterns in an efficient way and how they can be combined in an effective way. The method used to find additional occurrences of the motif class as defined by extracted similarity features is explained in Section III-D. Section III-E describes how patterns which do not belong to the motif class can potentially be processed into the model definition, while Section III-F explores how non-shape based features could be integrated in the model structure to identify more elaborate motif classes. Section III-G concludes with an overview on computational complexity of both processing verified patterns and the matching process.

A. DISTANCE MATRIX CALCULATION

The first step of defining the pattern model consists of calculating a distance matrix between two verified patterns. A distance matrix compares all subsequence pairs of two input patterns and gives a clear overview on which parts are similar to each other. The input patterns do not have to be the same size.

To construct a distance matrix between two verified patterns P_1 and P_2 with sizes n_1 and n_2, a window size $w_{1,2}$ is defined where $w_{1,2} < max(n_1, n_2)$ using Equation 1. This window size scales with the minimum of the two pattern sizes and the manually set size modifier m, to make sure enough detail is captured of either pattern, and has a minimum cap b to make it more noise robust. For the rest of this paper, the values set for m and b are 0.2 and 5 respectively, which were chosen experimentally.

$$w_{1,2} = max((min(n_1, n_2) * m), b) \tag{1}$$

The window is then used to extract all subsequences from P_1 and P_2 by scrolling over each of them with a step size of 1, extracting the subsequence at each step. This means extracted subsequences will overlap, giving a total of $n_1 - w_{1,2} + 1$ and $n_2 - w_{1,2} + 1$ subsequences respectively. By having overlapping subsequences, the ranges of similar parts can be defined much more precisely, as one point can have significant impact in case of outliers or level shifts. All pairs of subsequences are then compared with each other by computing the distance between them, constructing a $(n_1 - w_{1,2} + 1) \times (n_2 - w_{1,2} + 1)$ matrix. Typically a z-normalized distance measure D_{z-norm} is used to compute the distance measure between two subsequences, as defined in Equation 2, which describes the z-normalized distance between two subsequences S_1 and S_2, each of size w. In this equation, μ_1 and μ_2 denote the averages and σ_1 and σ_2 denote the standard deviations of their respective subsequence. The z-normalized distance measure allows each subsequence to be compared regardless of the level, noise, or outliers that occur in other parts of the input sequence.

$$D_{z-norm}(S_1, S_2) = \sqrt{\sum_{l=0}^{w-1} \left(\frac{S_{1+l} - \mu_1}{\sigma_1} - \frac{S_{2+l} - \mu_2}{\sigma_2} \right)^2} \tag{2}$$

As it does not take scale differences into account, this can produce big sequence differences, as shown in Figure 2. Therefore, a scale-aware distance measure is used to produce a more consistent result with what humans would perceive as similarity. The distance measure takes the relative scale difference as an additional factor, as calculated in Equation 3, caps it with a hyperparameter c, as shown in Equation 4, and merges it with the z-normalized distance using Equation 5, with w representing the subsequence size. The scale-aware distance value is bounded to $[0 - 1]$ and removes large scale differences on otherwise similar shapes from the distance matrix. The metric has already been used in previous work for discovering motifs, showing its benefit over z-normalized distance [6].

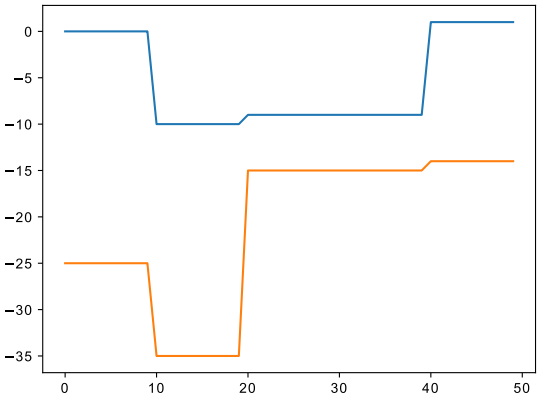

FIGURE 2. An example of two patterns that are very different to a human, but will generate a perfect match when comparing their scrolling windows with the z-normalized distance metric.

$$D_{scale}(S_1, S_2) = \frac{max(std(S_1), std(S_2))}{min(std(S_1), std(S_2))} - 1 \tag{3}$$

$$D_{scale-capped}(S_1, S_2) = \frac{min(D_{scale}, c)}{c} \tag{4}$$

$$D_{scale-aware}(S_1, S_2) = \frac{\sqrt{\left(\frac{D_{z-norm}(S_1,S_2)}{2\cdot\sqrt{w}}\right)^2 + D_{scale-capped}(S_1, S_2)^2}}{\sqrt{2}} \tag{5}$$

An example distance matrix output with this distance measure is shown in Figure 3. Using this matrix, similarities between the two input patterns become very clear as they generate regions with low distance values. Using a distance matrix also provides additional properties that are useful in building a matching model:

1) Temporal transformation of similarity: As pattern occurrences are not always the same length and as such, the pattern features that make every occurrence identifiable can shift with respect to each other at each occurrence. In the example presented in Figure 3, similarity is detected in the slope down and slope up, but they occur in different regions of the input subsequence. Using a distance matrix, they can both be detected and processed in the model.
2) Diagonal patterns: Prominent similar regions between the input subsequences generate diagonal patterns in the distance matrix where distance values are low. Extracting these diagonals provides a great primitive in building the model for finding pattern occurrences.

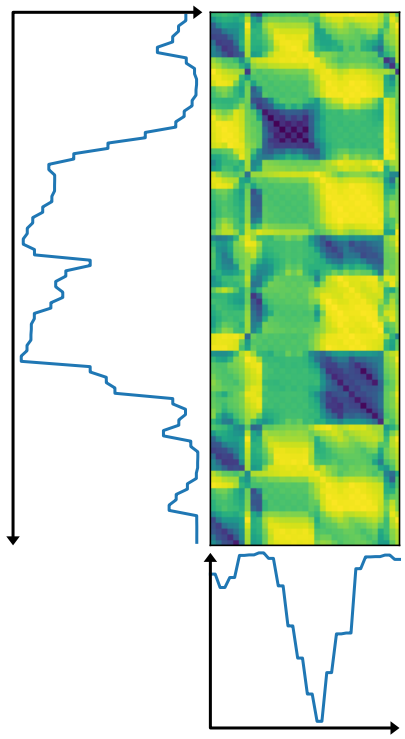

FIGURE 3. An example of a distance matrix calculated between two verified patterns. Low distance values are marked in dark blue, whereas green and yellow regions mark high distance values. Two similarities are noticeable when comparing these two patterns. Their slope down and their slope up generate heat points.

B. SIMILARITY PATH SEARCH

Using the calculated distance matrix as an input, a model is constructed that can find new occurrences based on the similarity between the verified patterns. The output is visualized by Figure 4, where the red marked regions depict the extracted similarity features of the verified patterns. To do this, diagonal patterns are first extracted from the matrix. The matrix values are converted into a discrete space 0, 1 using Equation 6, where v is the input matrix value and t is a threshold hyperparameter bounded to $[0-1]$. As the matrix values have a predefined range due to the $D_{scale-aware}$ metric, and the fact that scale differences are relative and therefore independent to the global scale and level of the patterns, a static value can be chosen that generally works well on any use case.

$$v_{bin} = \begin{cases} 1, & \text{if } v \leq t \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Using this binary matrix, diagonals are extracted by processing each diagonal line in the matrix, increasing the x and y coordinates with one each step, and grouping subsequent matrix indices where the binary value is 1 along the line. The process is described in Algorithm 1. For each diagonal, containing subsequent matrix index pairs $[d_1, d_2, \ldots, d_k]$ a cumulative score is also computed. This score describes how valuable a diagonal is to be included in the model. The score is calculated by reversing the distance values that fall in that diagonal, as described in Equation 7, where d_i represent the matrix indices and M the distance matrix containing distance values calculated using Equation 5. This means that a diagonal increases in value when its size increases or when the similarity within the diagonal is greater. A diagonal with a higher score has a higher chance of being included.

$$Score_{diagonal}(d) = \sum_{i=1}^{i \leq k} 1 - M[d_i] \tag{7}$$

Algorithm 1 Diagonal Extraction

M = Input matrix of dimensions $p \times q$ containing distance values
M_{bin} = Input binary matrix of dimensions $p \times q$
O = empty output list
for $i = -p + 1$, $i{+}{+}$, `while` $i < q$ **do**
  D = initialize empty list
  for $j = 0$, $j{+}{+}$, `while` $j < min(p, q)$ **do**
    $x = -1 * min(0, i) + j$
    $y = max(0, i) + j$
    if $x \geq p$ or $y \geq q$ **then**
      Break
    end if
    if $M_{bin}[x, y] = 1$ **then**
      if D is empty **then**
        $D = [[x, y], [x, y], 1 - M[x, y]]$
      else
        $D[1] = [x, y]$
        $D[2] += 1 - M[x, y]$
      end if
    else
      append D to O
      D = initialize empty list
    end if
  end for
end for
return O

The computed diagonals and their cumulative score are then used as the main primitives for finding the best matching model given the verified patterns. A warping path is constructed by finding the best combination of diagonals. To construct this path, an iteration is done over all extracted diagonals, selecting them as a potential starting point and chaining potential candidates together. The chain with the highest score is then selected as the warping path result. This is done via a recursive two step process. First, the potential candidates to add to the chain are selected by filtering out all diagonals that occur before the previous link in the chain. These cannot be added as it would potentially mean the same range on the verified patterns is selected twice, introducing ambiguity. After filtering, all potential candidates are iterated and the same process is repeated. When no candidates remain, the cumulative score of the selected diagonals in the warping path is computed and compared against other warping path candidates. The warping path with the largest score is then selected. The process is described in Algorithm 2, which uses Algorithm 3 as a recursive process to identify the best diagonals for each subsequent candidate.

Algorithm 2 Warping Path Calculation

D = Input diagonals as computed in Algorithm 1
$best_{score} = 0$
$best_{candidates}$ = empty list
for d `in` D **do**
    $score, candidates$ = Algorithm 3(D, d)
    if $score > best_{score}$ **then**
        $best_{score} = score$
        $best_{candidates} = candidates$
    end if
end for
reverse the order of $best_{candidates}$
return $best_{candidates}$

The warping path consists of a collection of non overlapping diagonals (or rather range pairs that don not overlap) and their position in the distance matrix. A visual example on this is illustrated in Figure 4. It provides an overview of what parts are important in identifying additional occurrences of the pattern. The parts that are not included in the warping path are to be ignored in the matching process, as they are not consistent between verified matches that the user provided.

The diagonals are then processed for matching by extracting the subsequences of the diagonal on both axes. This means each diagonal will result in two subsequences, for each of the two verified subsequences respectively. These range pairs are then grouped together and put in a chain of parts that have to match.

The positions of the extracted ranges are then used to construct additional temporal rules. These are needed to detect pattern occurrences at different sizes. As seen in Figure 4, the model needs to be able to match on both the small verified pattern and the large one. To achieve this, the gaps between the extracted ranges across each axis is taken as a temporal boundary where a diagonal match should occur.

Algorithm 3 Warping Path Recursive Function

D = Input diagonals
C = Candidate diagonal
$best_{score} = 0$
$best_{candidates}$ = empty list
for d `in` D **do**
    if $d[0], [0] > C[1], [0]$ and $d[0], [1] > C[1], [1]$ **then**
        $score, candidates$ = Algorithm 3(D, d)
        if $score > best_{score}$ **then**
            $best_{score} = score$
            $best_{candidates} = candidates$
        end if
    end if
end for
append C to $best_{candidates}$
$best_{score} += C[2]$
return $best_{score}, best_{candidates}$

Visualized in Figure 5, the fourth diagonal occurred 5 points after the third one along the x-axis, and 39 points along the y-axis. The extracted temporal rule will state that the fourth diagonal should match within $[5 - 39]$ points of the end of the third diagonal. This ensures size differences are allowed in the pattern occurrences while not being significantly different from what was already verified.

As a result, the model consists of a collection of ranges that have to match on an incoming query target together with delay constraints on what the temporal distance between each range is allowed to be. As the raw data of the similar ranges is kept, this results in a white-box model, meaning matching results can be explained through the extracted ranges, increasing human understanding on model behavior.

C. SCALING BEYOND TWO PATTERNS

Because the method described in Section III-A and Section III-B keeps the amount of verified patterns to two, the question remains on how to scale this method to additional input patterns. When additional verified patterns need to be added, there are two possibilities on constructing the model:

1) Destructive method: every additional pattern will increase the distance matrix dimensions in which the diagonals are extracted. This is making sure similarity is present in each verified input pattern the same way.
2) Constructive method: every new pattern will construct an additional model based on two input patterns and add it to the model. In other words, the extracted similarity is between two verified input patterns, ignoring cases where this similarity is not present in all verified patterns.

Both have their advantages and disadvantages. Starting with the destructive method, it reduces the amount of noise

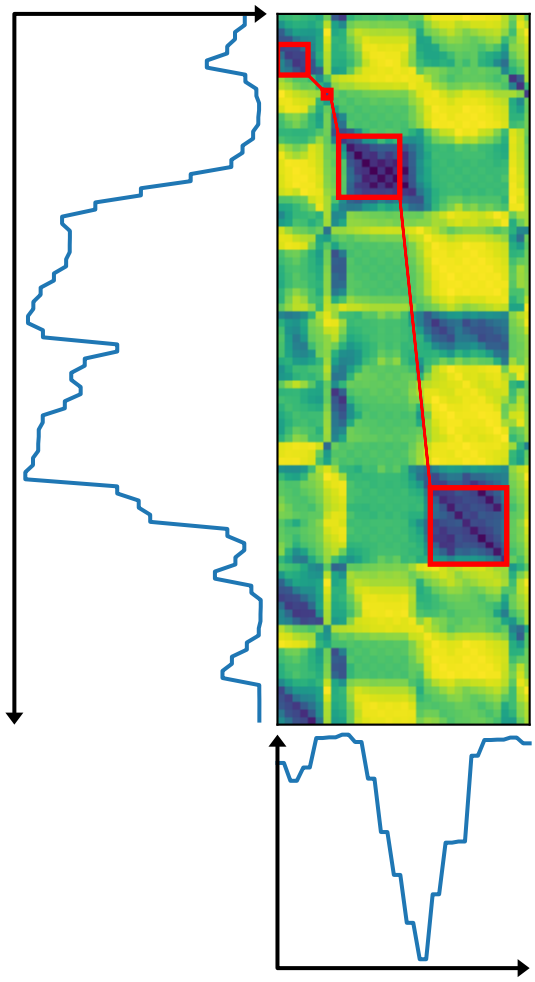

FIGURE 4. An example output of a warping path on a distance matrix between two verified patterns. Red rectangles are drawn around the selected diagonals and are connected on how the warping path is constructed. As shown, despite there being diagonals present in the bottom left corner, they are ignored in favour of a better diagonal combination. The middle portion of the longer pattern is ignored as similarity is not prominently detected here.

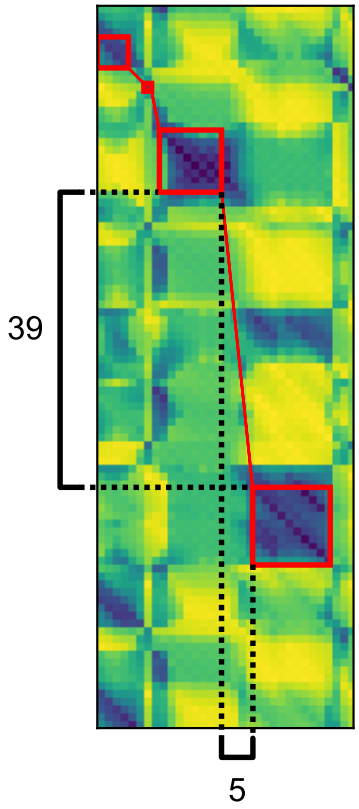

FIGURE 5. An example of how the delay range is calculated, by measuring the distance between the extracted diagonals along each axis, the temporal range is extracted. In this case, the third and fourth diagonal have a distance between 5 and 39 datapoints.

that can occur in the model significantly. This is because the odds of noise being consistently present in three or more patterns in the same position in the matrix is very small and decreases as patterns are added. The need for this benefit is visualized in Figure 6, where two spike patterns introduce some noise in the resulting model. Adding an additional verified spike removes this noise while preserving the spike. There are some disadvantages related to this model however. First, as every additional pattern increases the space needed to compute the model at an exponential degree, there is a big performance problem to deploy this in production environments where resources are scarce and the amount of input patterns can scale to infinity. Second, as the ranges in a model reduce with every additional pattern (in the best case scenario, they stay the same), prominent similarities could be lost in the process because there is a difference to one of the input patterns. This could lead to an overgeneralization of the model, where in the example displayed in Figure 4 for instance, this could mean only the level shift down is preserved, turning the model into an unintended form.

To mitigate these issues, a constructive approach could be used, where only pairwise models are calculated with every additional pattern. The process calculates every pairwise model with existing verified patterns and picks the one where the similarity ranges with respect to the new verified pattern is the largest, using Equation 8, where the new verified pattern P_x of length $len(P_x)$ is compared to verified pattern P_1, generating model PM. The number of extracted diagonals in the model, symbolized as nPM is then iterated over, calculating the sum of the length of each extracted range $len(PM[i])$. This sum is then normalized to the size of the input pattern nPM_x. Choosing the largest model ensures no model will be constructed from extremely different input patterns, generating an overgeneralized model in the process.

$$Score_{model}(P_x, P_1) = \frac{1}{len(P_x)} \sum_{i=0}^{i<nPM} len(PM[i]) \quad (8)$$

The constructive approach has no exponential memory increase problem, as only 2D matrices are calculated. It also will not permanently remove similarity ranges, as existing models remain untouched. The approach is susceptible to noise, because taking the largest model will have a higher

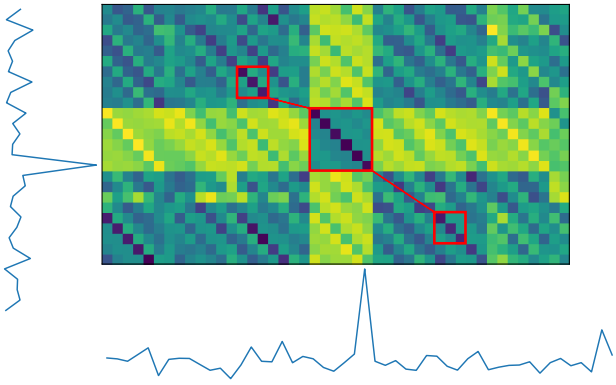

FIGURE 6. An example model where noise is present. The spike pattern is detected (note that each pixel in the matrix represents a window of five datapoints), but is combined with some patterns in the noise that are also similar. Other parts also exhibit similarity such as in the bottom left corner of the matrix. Increasing the number of input patterns will maintain the spike in the model, but the chance of the same noise pattern occurring in the same order to the spike decreases significantly.

chance of including noise as the number of verified patterns increases. A second problem is that the model size for matching increases to a larger extend than the destructive method, as a new pairwise model is introduced with every verified pattern compared to a refinement on an already existing model.

As both the destructive and constructive approaches have their strengths and weaknesses, a hybrid approach was designed to maximize overall accuracy.

The hybrid method works as follows. First, a pairwise model is calculated between the new verified pattern and all the patterns that are already in the model. With each pairwise model, the constructive approach model score, as described in Equation 8, is calculated to pick the best constructive model.

Before selecting the best constructive model, a destructive approach is calculated for each existing model. When the result of this destructive approach does not destroy a large amount of information (i.e., a lot of similarity is still visible between all patterns in the model), it is selected over the constructive approach, as this is an indication of coincidental similarity that was captured before is now filtered out.

Calculating the exact destructive model does not scale well with the amount of input patterns, as each additional verified pattern increases the number of dimensions of the similarity distance matrix that needs to be calculated (i.e. 2D matrix for two verified patterns, 3D for three patterns, etc.). This exponential increase in both memory and computations is too costly to maintain in any production environment. To mitigate this, an educated guess can be computed by calculating the 2D distance matrices between each pattern pair in the destructive model and merging any overlapping ranges from the extracted similarity. This results in only 2D matrix calculations needing to be computed, which significantly reduces memory and number of computations needed.

Merging overlapping similarity ranges in the destructive model is done in two steps. First, the overlap is calculated of each extracted range per pattern. In other words, computing a destructive approach for patterns P_1, P_2 and P_3, it will generate three 2D matrices, where each input pattern will have two extracted range results (e.g. P_1 will have similarity ranges to P_2 and to P_3). Overlapping the range result of each pattern will show an educated guess on what similarity is present in the 3D distance matrix. Second, the overlapping ranges of each input pattern is translated in order to compute the overlapping ranges of all patterns combined. If P_1 is very similar to P_2 and P_3, but P_2 and P_3 are not as similar to each other, the resulting overlapping range of P_1 has to reflect that, as all three input patterns have to match on the extracted range altogether. After translating the overlapping range to all other patterns, the overlap is then calculated again so that the range of each pattern is consistent to what a 3D matrix result would output.

Calculating the destructive approach like this removes the need to calculate a 3D distance matrix, reducing the space and time complexity from $O(s^n)$, where s denotes the pattern size and n the number of patterns to $O(s^2 n)$, making it much more scalable. Worth noting however is that the result is still an educated guess that is not always equal to the result of a true 3D matrix calculation, as the warping path is chosen before computing the overlapping ranges.

After calculating the destructive approach to each model, a decision is then made on whether it should take priority over picking the best constructive approach model to the new verified pattern. This is done by checking if a lot of range data is thrown away using a destructive approach or not. In other words, when a large portion of the already present range data is not preserved in the overlapping range, it probably means significant similarities from the old model are discarded, which is not desired in most cases. Therefore, if the amount of discarded data is too high, the best constructive model is used instead. The decision formula is described in Equation 9, where the overlapped model PM' contains nPM' ranges and the range size at index i is denoted by $len(PM'[i])$. The old model PM has nPM ranges, where the size of the range at index i is denoted by $len(PM[i])$. A hyperparameter $t_{destructive}$ in the range $[0-1]$ is introduced as a decision boundary.

$$Des = \frac{\sum_{i=0}^{i<nPM'} len(PM'[i])}{\sum_{i=0}^{i<nPM} len(PM[i])} \geq t_{destructive} \tag{9}$$

D. MATCHING REFINED PATTERNS

The output of the model creation is a now a list of models. Each model includes the diagonals of parts that have to match, with each diagonal including the subsequences of all the verified patterns in the model. Inbetween the ranges, a temporal rule is also present, which is a delay range in which the following diagonal should match. A visual overview is illustrated in Figure 7.

Using this model to find similar occurrences in a given time series is done as follows. First, an iteration is done over every index in the time series. For each iteration, all models in the model list are checked whether they match on the specified

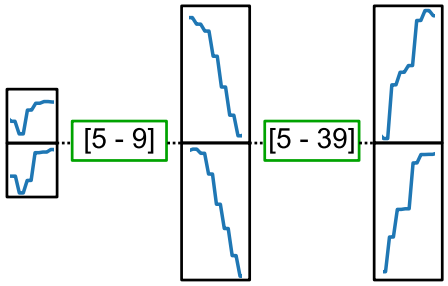

FIGURE 7. An example model to illustrate its structure. In this example, three diagonals were extracted. Between the extracted diagonals, a delay range is saved as a temporal rule between them. Between the first and second diagonal, a delay of 5-9 steps is required, and between the second and third diagonal, a delay of 5-39 steps is required.

index. If a match is found, the largest match is chosen and put in the occurrence list.

Matching a model is done by checking if all diagonals match within the set temporal rules between them. A diagonal matches if the minimum distances of the target subsequence to every subsequence in the diagonal is below a set threshold. The distance measure used is the scale-aware measure defined in Equation 5.

Checking all diagonals in a recursive process is straightforward, but can potentially lead to a lot of duplicate calculations. To prevent this, a distance profile is calculated for each diagonal, generating a distance matrix with a width equal to the number of diagonals in the model that need to match and a height of the maximum model size, which is equal to the maximum length of the. Each value in the distance profile is the minimum distance value of each diagonal range compared to the target subsequence.

Using this matrix, a warping path can then be computed that adheres by the temporal rules set between each diagonal. the best warping path can the computed similarly as Algorithm 2 but computes the score based on the match size, meaning the larger match will always win. The process of matching a model on a time series index is explained in Algorithm 4 and uses Algorithm 5 for recursively matching every extracted diagonal within the temporal constraints.

E. NEGATIVE VERIFIED PATTERNS

Until this point, only positive patterns have been discussed, meaning every verified pattern is typically an occurrence that was not found in the current model that the user wants to be included in future matching. Because the model will occasionally have false positives, including the ability for a user to exclude those and occurrences like those from the model can be useful.

This can be done by introducing a new list of models that holds the shape information of all patterns that the user does not want, also called negative patterns. The models in this list are defined in the same way as as positive patterns, as discussed in Section III-A, III-B and III-C.

Algorithm 4 Model Matching

PM = Input pattern model comprising of diagonals PM_r and delay ranges PM_d
T = Input time series
i = Input index of the time series that needs to be checked
th = Threshold
dis = minimum distance between ranges in $PM_r[0]$ and $T[i : i + len(PM_r[0]) - 1]$ using Equation 5
max match size = maximum size input time series of PM
if $dis \geq th$ **then**
    Return false, [-1,-1]
end if
POI = empty list of $len(PM_r) - 1$ empty points of interest lists
for $x = 0$, $x{+}{+}$, `while` $x < len(POI)$ **do**
    for $y = min(PM_d[0], [0])$, $y{+}{+}$, `while` $y <$ `max match size` **do**
        dis = minimum distance between ranges in $PM_r[x + 1]$ and $T[i + y]$ using Equation 5.
        if $dis < th$ **then**
            Add y to $POI[x]$
        end if
    end for
end for
$range_{best} = -1$
for y in $POI[0]$ **do**
    if $y \leq PM_d[0], [1]$ **then**
        r = Algorithm 5$(POI[1:], y + len(PM_r[0], [0]) + PM_d[0], [0], PM_d[1:], PM_r[1:])$
        $range_{best} = max(r, range_{best})$
    else
        Break
    end if
end for
if $range_{best} > -1$ **then**
    Return true, $[i, i + br]$
end if
Return false, [-1,-1]

The difference lies in how the model is matched. As of now, there are two input model lists, one for positive patterns and one for negative patterns. When a match is found in the positive model list, as described in Section III-D, the same match range can be checked for a match in the negative model list. If there is a negative match, the match result is ignored. The process is described in Algorithm 6.

F. NON SHAPE BASED DISTANCE MATRICES

As the method is based on the ability of comparing subsequences to each other, it is not limited to refining patterns based on their exact shape. Any metric which can be calculated from subsequences and compared in a distance measure can be used. An example is shown in Figure 8, where the amount of noise is calculated and compared using the

Algorithm 5 Model Matching Recursive Process

POI = Input list of points of interest
d = Input current delay
PM_d = Input list of delay ranges
PM_r = Input list of diagonals
if $len(POI) == 0$ **then**
    Return d
end if
$range_{best}$ = -1
for y in $POI[0]$ **do**
    if $y - d \leq PM_d[0], [1]$ and $y - d \geq PM_d[0], [0]$ **then**
        r = Algorithm 5(POI[1], $y + d + len(PM_r[0], [0])$, $PM_d[1:], PM_r[1:]$)
        $range_{best} = max(r, range_{best})$
    end if
end for
Return $range_{best}$

Algorithm 6 Negative Matching

PM = Input list of positive verified patterns
NM = Input list of negative verified patterns
T = Input time series in which occurrences need to be found
O = empty output list
for i `in` T **do**
    for pmi `in` PM **do**
        $Match_p, Range_p$ = Algorithm 4(pmi, T, i)
        if $Match_p$ **then**
            $Match_n$ = false
            for nmi `in` NM **do**
                $Match_n, Range_n$
                = Algorithm 4(nmi, T, i)
                if $Match_n$ **then**
                    Break
                end if
            end for
            if $Match_n$==false **then**
                add $Range_p$ to O
            end if
            Break
        end if
    end for
end for
return O

scale metric described in Equation 4. Here, a similar noise increase is visible on both patterns. By changing the metric and therefore the definition of similarity, the model is capable of detecting both noise amounts as a similarity feature and will match on similar noise increases as a result. This is a simple example however and more complex metrics to define the values in the distance matrix are out of scope of this paper. Other work has been done that explores more elaborate ways on extracting key features in subsequences which can be used in this approach [8].

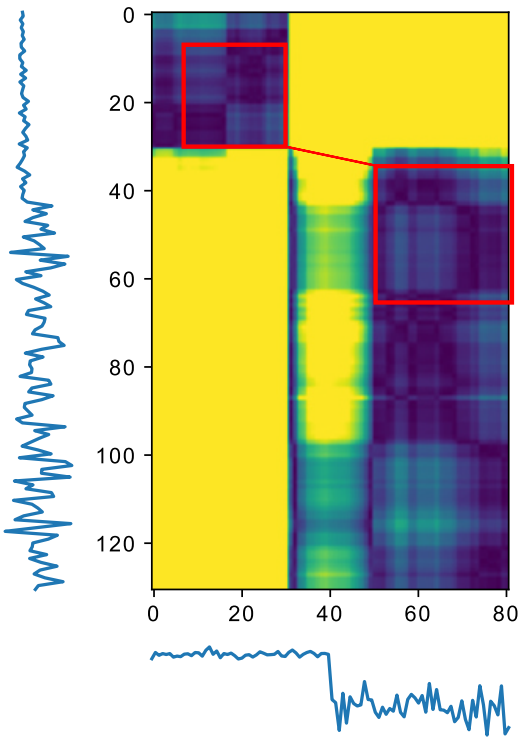

FIGURE 8. An example of a generated model using a non shape metric. The distance matrix compares the amount of noise in subsequences instead of exact comparisons. The resulting model focuses on the two noisy areas, ignoring the level shift in one of the models, and will match on sequences that experience a similar noise increase.

By using other metrics than exact shape comparisons, identifying occurrences of complex patterns in more challenging environments becomes possible. Because of the flexibility of the distance matrix based model creation, the proposed method could therefore potentially be deployed in a wide range of different use cases.

G. COMPLEXITY

The time complexity of computing the model with an additional verified pattern is influenced by two main factors. First, the size of the patterns s will define the size of each matrix that has to be computed, while the total amount of patterns n will define how many matrices will have to be computed. As such, the model creation has a time complexity of $O(s^4n^2)$ in worst case, as the number of diagonals that can be included in the warping path is also tied to s. In terms of space complexity, the model needs $O(s^2)$ memory to compute a new model, as each distance matrix can be discarded when the relevant diagonals are extracted.

For matching, the time complexity needed is tied to the number of patterns stored n, the size of each pattern s and the size of the target time series t. The time complexity can be defined as $O(s^2nt)$ in worst case. The diagonal path for each model cannot exceed s, as it would otherwise lead to

overlapping diagonals. However, due to the temporal rules set, the amount of matches computed for each diagonal scales with the total pattern size. The space complexity can be described as $O(sn + t)$, as the model size scales linearly with every added verified pattern and it's size.

In production environments, memory tends to be the first limitation in maintaining real time monitoring models. As such, the proposed algorithm scales this linearly during matching cycles and only spikes to $O(s^2)$ when a manual intervention is done in the form of adding verified patterns. The time complexity also follows this linear trend, with real time monitoring deployments only requiring $O(sn)$ calculations per time step in worst case. The time complexity does see a big spike when adding verified patterns. However, because adding verified patterns is a manual task and as such is processed offline, this should not pose an issue in most production environments.

IV. EVALUATION

To evaluate the method, several time series are given as input with each containing multiple pattern occurrences. The performance metrics consist of precision and recall scores, which are described in equations 10 and 11, where *TP* and *FP* depict the number of true positives and false positives during matching respectively and *FN* the number of false negatives. With each time series, a starting verified pattern range is defined. Initial matches are calculated by computing a distance profile over the time series using the scale-aware distance metric, described in Equation 5, after which the matches are extracted by applying a matching threshold. The matching threshold and the scale cap parameter were set to 0.3 and 1.5 respectively, values chosen experimentally. As described in previous work, the scale-aware distance measure results in less accidental matches, increasing the overall results compared to z-normalized Euclidean distance [6].

$$precision = \frac{number\ of\ TPs}{number\ of\ TPs + number\ of\ FPs} \tag{10}$$

$$recall = \frac{number\ of\ TPs}{number\ of\ TPs + number\ of\ FNs} \tag{11}$$

Feedback is then given to refine the model into improving the recall and precision of the matching. The number of feedback samples required to improve the result is then reviewed to check the effectiveness of the algorithm.

The datasets used include a synthetic benchmark, a real world dataset provided by Skyline Communications and a publicly available Human Motion Primitives dataset [25].

A summary of the results are shown in Table 1, where precision and recall scores are given for both distance profile based matching as well as feedback refined matching by using the proposed method. The number of additional feedback samples to achieve the scores is also given.

TABLE 1. Result summary of the proposed feedback method. Precision and recall scores are shown for distance profile based pattern matching and the proposed feedback method. It also shows the amount of additional feedback samples given for the refined result.

	Distance profile (precision - recall)	Feedback method (precision - recall)	Feedback samples given
Synthetic dataset	100% - 25%	100% - 100%	2
Skyline dataset	100% - 47%	100% - 100%	1
HMP dataset	100% - 38%	100% - 85%	2

A. SYNTHETIC DATASET

The first time series in the synthetic dataset, visualized in Figure 9, contains several block patterns in an otherwise noisy signal. The block signals differ in length, ranging from 42 to 147 datapoints. As such, calculating the distance profile on the first level shift only returns one match. Providing feedback by adding the second range increases the number of occurrences found from one to five matches, only mismatching the largest and two smallest patterns. Looking at the pattern data, the pattern identified consists of the level shift up and the level shift down being identified. A perfect recall score is achieved by tagging the largest and smallest pattern occurrence. Doing so does not increase the amount of models in the model list, but extends the existing model by increasing the delay range. By only tagging the smallest and largest patterns in the list, the perfect precision and recall score can be achieved with only two feedback samples.

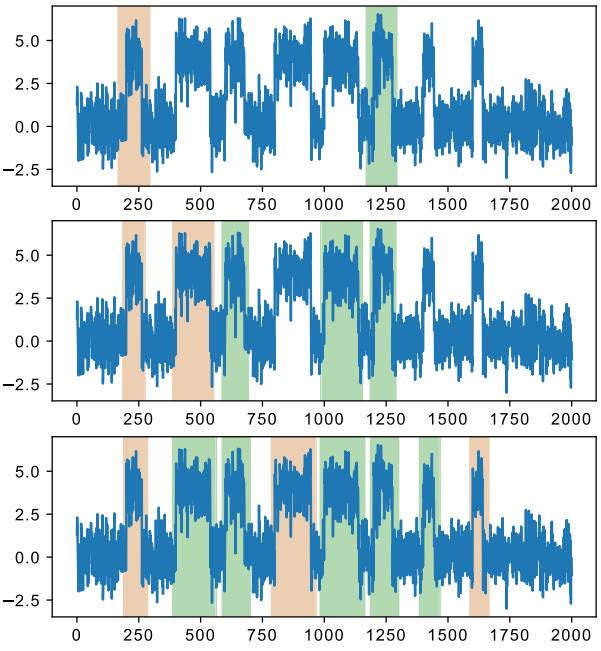

FIGURE 9. Results of the synthetic dataset, containing several block shapes with different sizes. Orange marked regions define model input, green marked regions together with orange marked regions show all identified matches. In this case, three input ranges are needed to get a perfect matching result.

Worth noting is that for optimal results, the user has to label the most extreme cases, as they will have a higher impact on the delay ranges, incorporating more matches in the process.

B. SKYLINE DATASET

The Skyline dataset consists of a real world memory KPI on a production server. It includes a daily pattern during a backup cycle that occurs daily. The pattern length is influenced by the amount of data being backed up, with differences ranging between 20 (around 1.5 hours in time) to 80 datapoints (around 6.5 hours in time). There are also shape differences between occurrences, as some generate a spike up in the middle, while others do not. The data and results are visualized in Figure 10.

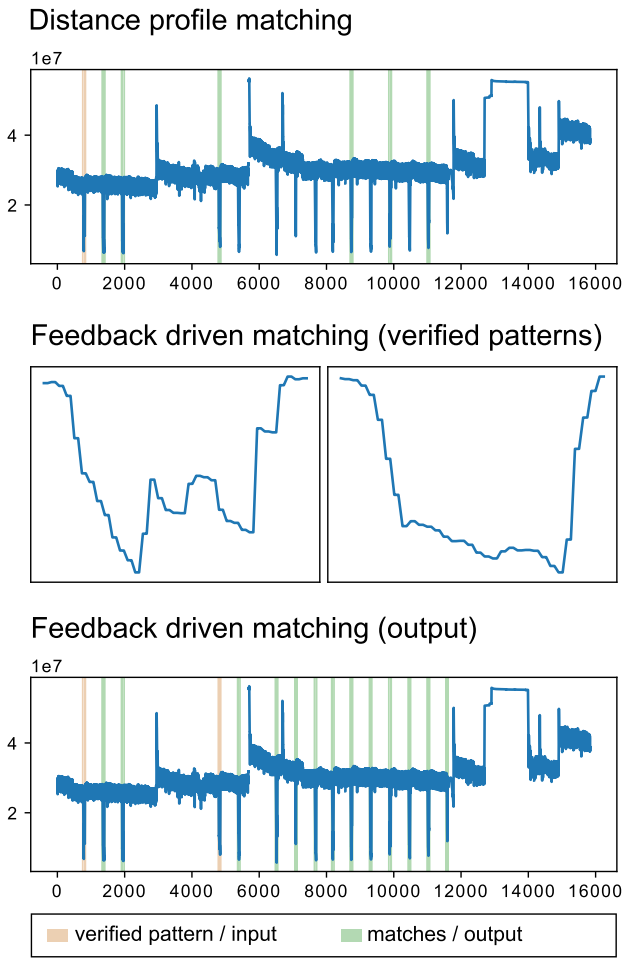

FIGURE 10. Results of the Skyline dataset. The datasets contain a daily pattern depicting a backup cycle, which cannot be fully matched using a distance profile method on the middle left subsequence. The matches are marked in green, the source pattern is marked in orange. Adding the middle right pattern as a feedback sample results in a complete match, as shown at the bottom.

A distance profile calculation is only able to identify 7 out of 15 occurrences. Generating a model with one additional occurrence increases this to 15 occurrences without introducing any false positives.

C. HUMAN MOTION PRIMITIVES DATASET

The human motion primitives dataset [25] contains accelerometer data on several activities such as climbing stairs, combing hair, brushing teeth, etc. While most of these generate patterns that can be found effectively with just a distance profile calculation, some examples generate more noisy patterns. For the first example, a time series from the combing hair dataset is used where a distance profile is only able to find 10 out of 26 movements. When providing the model with one additional sample however, this jumps to 18. Three feedback samples provides a model that is capable of detecting 22 out of 26 occurrences without triggering false positives, a considerable improvement.

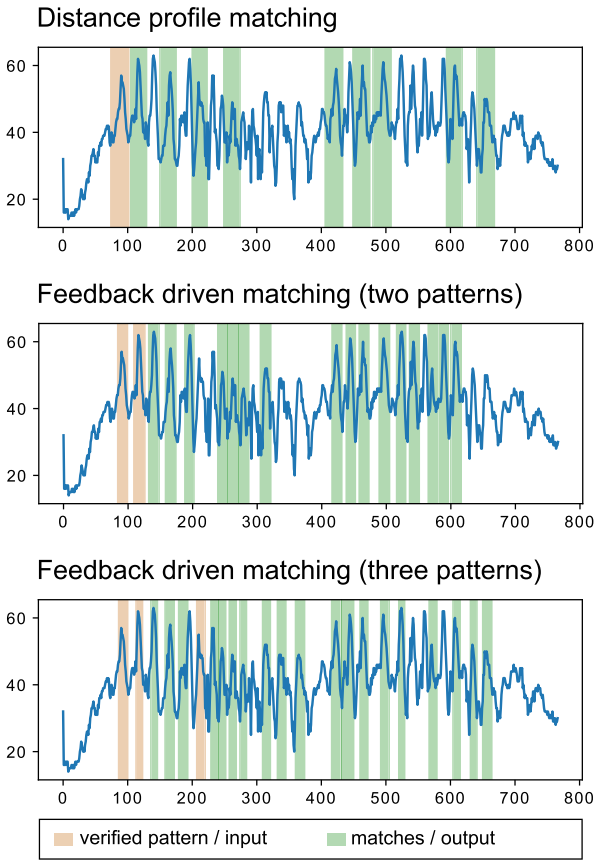

FIGURE 11. Results of a combing hair time series in the HMP dataset. The dataset contains 26 repeating patterns. Using a distance profile is not sufficient enough to detect all occurrences, only matching on 10 of them. Providing just 2 additional feedback samples improves this result to 22 without generating false positives.

V. CONCLUSION AND FUTURE WORK

This paper proposed a novel feedback driven way of finding pattern occurrences in time series data in a user convenient way. It is able to identify key similarity features in subsequences with very limited sample data from which an improved matching model is built. The technique is able to find motif occurrences with varying lengths and can filter out expected noise or shape deviations. Due to its white

box design, matching can be explained in a human readable fashion, giving the user more control on what characteristics are being monitored. The method was tested on synthetic and real world data, showing promising results when both temporal size and shape differences occur within the same pattern, as is the case in the Skyline dataset.

While the method works generally well and shows a lot of potential, there are some limitations that need to be addressed in future work. First, while providing negative feedback by maintaining a negative model list that takes priority over the positive models can work in some cases, it opens the door where the model can be permanently damaged when a negative match includes all the identifiable features of the positive models. As users will presumably limit their interaction by just tagging ranges without considering the models inner workings, there is a real possibility of this damage occurring. Future work is needed to provide a more user proof version of negative feedback and make the model widely deployable on production environments. One approach can be to match negative samples on the positive models, identifying how key positive features overlap with the negative sample and filtering those out of the matching of the negative model entirely. This way the negative model prioritizes features that makes them unique to the entire model list instead of just focusing on the negative models.

Second, while significant performance issues did not occur during testing, the fact that the model keeps getting larger as new samples are inserted could pose a problem in environments where a lot of operators are refining a single model deployed in a complex use case. To address this, future research can be done on implementing a compaction routine that can run in the background every so often, which could identify similar features throughout the model and aggregate them to make both matching and refining faster and future proof. Performance improvements can also be made on the warping path search, which now has a computational complexity that scales quadratic with every additional diagonal. Setting up a cumulative distance matrix similar to DTW in the diagonal feature space could provide a solution for this [26], but was not researched in this paper.

Lastly, non shape based metrics are only mentioned briefly and should be researched further to verify the feasibility in this model structure. While they are theoretically possible, practical results may vary based on the metric and model hyperparameters used.

While some limitations remain that have to be researched further, it brings current motif discovery algorithms a step closer to deployment in production environments by improving their reliability long term.

REFERENCES

[1] S. Torkamani and V. Lohweg, "Survey on time series motif discovery," *WIREs Data Mining Knowl. Discovery*, vol. 7, no. 2, p. e1199, Mar. 2017.

[2] E. J. Keogh and M. J. Pazzani, "Relevance feedback retrieval of time series data," in *Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* New York, NY, USA: Association for Computing Machinery, Aug. 1999, pp. 183–190.

[3] 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.

[4] 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.

[5] 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] M. Van Onsem, V. Ledoux, W. Mélange, D. Dreesen, and S. Van Hoecke, "Variable length motif discovery in time series data," *IEEE Access*, vol. 11, pp. 73754–73766, 2023.

[7] Y. Zhu, Z. Zimmerman, N. S. Senobari, 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. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 739–748.

[8] S. Imani, S. Alaee, and E. Keogh, "Putting the human in the time series analytics loop," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2021, pp. 635–648.

[9] 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.

[10] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: A novel symbolic representation of time series," *Data Mining Knowl. Discovery*, vol. 15, no. 2, pp. 107–144, Aug. 2007.

[11] Y. Gao and J. Lin, "Efficient discovery of time series motifs with large length range in million scale time series," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2017, pp. 1213–1222.

[12] O. Gold and M. Sharir, "Dynamic time warping and geometric edit distance: Breaking the quadratic barrier," *CM Trans. Algorithms (TALG)*, vol. 14, no. 4, pp. 1–17, 2018.

[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, É. Fromont, and J. Odobez, "Unsupervised interpretable pattern discovery in time series using autoencoders," in *Proc. Joint IAPR Int. Workshop*, Mérida, Mexico. Springer, Dec. 2016, pp. 427–438.

[15] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Mining Knowl. Discovery*, vol. 31, no. 3, pp. 606–660, May 2017.

[16] N. Mohammadi Foumani, L. Miller, C. Wei Tan, G. I. Webb, G. Forestier, and M. Salehi, "Deep learning for time series classification and extrinsic regression: A current survey," 2023, *arXiv:2302.02515*.

[17] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Müller, "Deep learning for time series classification: A review," *Data Mining Knowl. Discovery*, vol. 33, no. 4, pp. 917–963, Mar. 2019.

[18] Z. Wang and T. Oates, "Imaging time-series to improve classification and imputation," 2015, *arXiv:1506.00327*.

[19] S. Köknar-Tezel and L. J. Latecki, "Improving SVM classification on imbalanced time series data sets with ghost points," *Knowl. Inf. Syst.*, vol. 28, no. 1, pp. 1–23, 2011.

[20] R. Huerta, S. Vembu, M. K. Muezzinoglu, and A. Vergara, "Dynamical SVM for time series classification," in *Pattern Recognition*, A. Pinz, T. Pock, H. Bischof, and F. Leberl, Eds., Berlin, Germany: Springer, 2012, pp. 216–225.

[21] E. Aasi, C. Ioan Vasile, M. Bahreinian, and C. Belta, "Classification of time-series data using boosted decision trees," 2021, *arXiv:2110.00581*.

[22] V. Shalaeva, S. Alkhoury, J. Marinescu, C. Amblard, and G. Bisson, "Multi-operator decision trees for explainable time-series classification," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations*. Berlin, Germany: Springer, 2018, pp. 86–99.

[23] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Inf. Sci.*, vol. 239, pp. 142–153, Aug. 2013.

[24] D. Rajput, W.-J. Wang, and C.-C. Chen, "Evaluation of a decided sample size in machine learning applications," *BMC Bioinf.*, vol. 24, no. 1, pp. 1–17, Feb. 2023.
[25] B. Bruno, F. Mastrogiovanni, and A. Sgorbissa. (2014). *Public Dataset of Accelerometer Data for Human Motion Primitives Detection*. [Online]. Available: https://github.com/boberle/bibtex_entry_generator
[26] M. Müller, "Dynamic time warping," in *Information Retrieval for Music and Motion*, 2007, pp. 69–84.

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 main research interests include 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 mostly focuses on time series insights mining. Her research was mostly 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 was 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 was 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 from the Department of Information Technology, Ghent University, with a focus on efficient service management in healthcare. She is currently an Associate Professor of semantic intelligence with IDLab, imec, Ghent University. 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.