Discovering and Labeling Power System Events in Synchrophasor Data with Matrix Profile

Jie Shi, Nanpeng Yu, Eamonn Keogh
University of California Riverside
Riverside, California 92521
Email: nyu@ece.ucr.edu

Heng (Kevin) Chen
Commonwealth Edison
Oakbrook Terrace, Illinois 60181
Email: heng.chen@comed.com

Koji Yamashita
Michigan Technological University
Houghton, Michigan 49931
Email: kyamashi@mtu.edu

***Abstract*—An increasing number of phasor measurement units (PMUs) are being installed to improve power systems' reliability and visibility throughout the world. Due to the high sampling speed, PMUs generate a large volume of streaming synchrophasor data. This huge dataset calls for robust and efficient data analytic tools to discover and label system events, which will greatly enhance the stability of power systems. In this work, we introduce a novel event discovery and labeling framework based on matrix profile. This framework is model-free, fast, scalable, and only requires one user-defined parameter. Since matrix profiles are built by measuring the similarities between subsequences of a time series, our approach has great potential in automatically labeling the system events in the synchrophasor data. Case studies are carried out on real-world PMU data to validate the effectiveness of the proposed framework.**

***Index Terms*—PMU, synchrophasor data, anomaly detection, matrix profile, time series.**

I. Introduction

Phasor measurement units (PMUs) are devices that provide time-synchronized measurements of different variables in a power grid. PMU data are also called synchrophasor data and are taken at high temporal resolutions such as 30 to 60 records per second [1]. This sampling speed is a significant improvement over the traditional supervisory control and data acquisition (SCADA) system which takes measurements every 2 to 4 seconds. Thanks to the fast streaming speed and high quality of synchrophasor data, the broad deployment of PMUs has greatly enhanced power systems' visibility and reliability. For example, PMUs have been utilized to improve wide-area situational awareness [2], state estimation [3], system protection [4], and control [5]. Meanwhile, the high sampling frequency of PMUs brings the grid operators unprecedented quantities of data describing the system conditions with high temporal resolution. This huge streaming dataset calls for robust and efficient data analytic tools to discover hidden information in a timely manner. Valuable data-driven applications can be built upon these tools, thus benefiting the power system operation. In this work, we investigate the utilization of a novel data analytic tool called matrix profile (MP) to discover and label system events in real-world PMU data.

Event (fault, anomaly, or disturbance) discovery based on PMU data has been extensively studied in the past decade. The existing approaches can be divided into model-based methods and model-free methods. Many of the model-based approaches develop estimates of the power system states based on the given model information. An anomaly is detected if the differences between raw measurements and the estimates are beyond certain thresholds. See [6] for an example. The performance of model-based methods heavily relies on the accuracy of model parameters. This dependency renders them less effective in real-world applications where model information is typically noisy [7]. To the best of the authors' knowledge, most commercial software does not use model-based methods for anomaly detection. In addition, these approaches do not provide useful information for further labeling purposes.

As a consequence, a lot more research efforts paid attention to the model-free analyses. The model-free methods can be further categorized into two classes, which are signal processing based approaches and machine learning based approaches. The key idea of signal processing based approaches is to monitor the coefficients of certain basis functions obtained through, for example, wavelet analysis [8], [9], [10]. A system event is detected if the ranges of certain coefficients (or some indices calculated from these coefficients) exceed the thresholds. This mechanism can work well with proper selection of wavelets, time and frequency resolution, threshold, etc. However, power system events are complex and diverse, making it difficult to select proper settings for different scenarios. Still, many researchers are working on these methods because they extract features from the raw event data that can be very useful for further clustering and labeling (classification).

The machine learning based approaches can be further divided into supervised learning methods and unsupervised learning methods. Typical supervised learning methods used in anomaly detection include decision trees [11], K-nearest neighbor [12], support vector machines [13], extreme learning machines [14], and artificial neural networks [15]. These works usually discover systems events and classify them into different groups, which require a sufficient amount of labeled training data. In practice, however, there are two obstacles. First, most PMU datasets are still weakly labeled since manually labeling them would yield considerable labor cost. Moreover, the low frequency of system events over the entire time horizon makes the training data extremely unbalanced. These two issues need to be addressed before supervised learning methods can be effectively employed in real-world applications. The unsupervised learning methods,

on the other hand, do not require any labeled data. Previous works in this category include characteristic ellipsoid based detection [16], principal component analysis (PCA) with linear regression [17], ensemble of three basic detectors [18], time series modeling [7], and K-means clustering [19]. However, these methods also have limitations. [16] is difficult to apply in an online environment due to the requirement of solving a non-convex optimization problem. [17], [18], and [7] only detect the anomalies without directly providing sufficient information for labeling them. [19] requires the specific number of clusters to be predefined.

An ideal event discovery and labeling algorithm for PMU data should be model-free, parameter-free, and capable of mining weakly labeled data. In this paper, we propose a novel event discovery and labeling framework based on matrix profile to address the above mentioned research needs.

The advantages of the proposed approach are summarized as follows:

- Our framework is model free, almost parameter-free, and capable of mining weakly labeled data.
- Our framework has great potential in automatically labeling the power system events in PMU data.
- Our framework is fast, scalable, and can be implemented in an anytime fashion.
- For streaming PMU data, our framework supports efficient incremental learning, which allows online implementation.

Equipped with all these merits, our framework has great potential in discovering and labeling the power system events with a large volume of streaming PMU data.

The rest of the paper is organized as follows: Section II describes the background of a matrix profile algorithm and its mechanism in anomaly detection. Case studies are carried out in Section III to validate the effectiveness of the proposed framework. The conclusion is stated in Section IV.

II. Background of Matrix Profile

In this section, we present the basic background of a matrix profile and its mechanism of event discovery and labeling in PMU data. Knowing that a general matrix profile is defined on two separate time series [20], we only focus on a special setting where both the time series are exactly the same. The resulting matrix profile is formally called self-join matrix profile. Without raising any confusion, we will hereby refer to self-join matrix profile as matrix profile, omitting "self-join".

Briefly speaking, the matrix profile of a time series is a vector of Euclidean distances between all z-score scaled subsequence of that time series and their nearest neighbors. This statement will become more concrete and clear after we introduce the necessary notations and supporting definitions. The intuition behind a matrix profile is simple and straightforward, i.e., measuring the similarities between different parts in a time series. We elaborate on the specific concept and details of a matrix profile in the following subsection.

A. Notations and Definitions

This work focuses on mining time series data. We begin by giving a formal definition of time series.

Definition 1. A time series T is defined as a sequence of real-valued numbers indexed in time order.

The properties of a complete time series are not our primary concern. We are interested in studying the patterns of relatively small segments of a time series. We call these segments subsequences.

Definition 2. A subsequence $T_{i,m} \in \mathcal{R}^m$ of a time series T is a continuous subset of T, which starts at the ith element and ends at the $(i+m-1)$th element of T.

There can be a large number of different subsequences in a given time series, especially when m is much smaller than the length of original time series. To better describe the set of all these subsequences, we introduce the following definition.

Definition 3. An all-subsequences set $\mathcal{S}^A$ of a time series T is defined as an ordered set of all possible subsequences in T. Specifically, these subsequences are sorted in ascending order with respect to the indices of their starting elements. $\mathcal{S}^A = \{T_{1,m}, T_{2,m}, \cdots, T_{n-m+1,m}\}$, where m is the length of any subsequence and n is the length of original time series.

For any subsequence in a time series, we can calculate its distance to all other subsequences in the same time series. The results are stored in a vector that is named as distance profile. Below is the formal definition of distance profile.

Definition 4. A distance profile $D_i^T \in \mathcal{R}^{n-m+1}$ between a time series T and its subsequence $T_{i,m}$ is a vector that stores $d(T_{i,m}, T_{j,m}), \forall j \in \{1, 2, \cdots, n-m+1\}$, where $d(\cdot)$ calculates the Euclidean distance between z scores of two subsequences.

A subsequence $T_{j,m}$ is defined as the nearest neighbor of subsequence $T_{i,m}$ if $d(T_{i,m}, T_{j,m}) = \min(D_i^T)$. Note that closely located subsequences are very similar in shape, thus they typically have the lowest distances. Therefore, an exclusion zone of length m is set up to filter out these trivial candidates. We define the subsequences in this exclusion zone as trivial neighbors of $T_{i,m}$.

Definition 5. A subsequence $T_{j,m}$ is defined as a trivial neighbor of $T_{i,m}$ if $i - \frac{m}{2} \leq j \leq i + \frac{m}{2}$.

Now we are ready to introduce the formal definition of matrix profile as follows:

Definition 6. A matrix profile $P^T \in \mathcal{R}^{n-m+1}$ of time series T is a vector that stores the Euclidean distances between each subsequence $T_{i,m}$ and its nearest non-trivial neighbor. Specifically, $P^T = [nn_{nt}(D_1^T), nn_{nt}(D_2^T), \cdots, nn_{nt}(D_{n-m+1}^T)]$, where $nn_{nt}(D_i^T)$ returns the smallest distance between $T_{i,m}$ and its non-trivial neighbors.

This vector of distance values is named matrix profile because one can obtain it by calculating all the distances

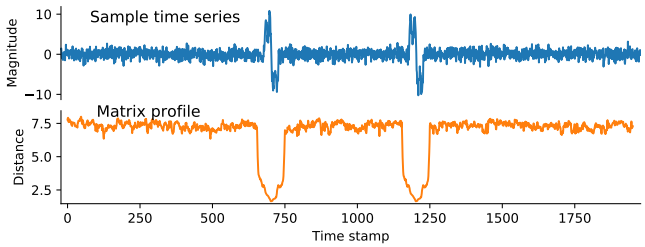

Fig. 1: Sample time series and its matrix profile.

between any pair of subsequences in a time series and put the results in a distance matrix, where the matrix profile is derived by extracting the smallest distance that is not between trivial neighbors out of each row. This simple brute force approach has a time complexity of $O(n^2m)$ and large overhead caused by z-score scaling[1]. Fig. 1 shows a sample time series of 2,000 data points and its matrix profile calculated with $m = 50$. In the next subsection, we briefly introduce a more efficient algorithm called scalable time series anytime matrix profile (STAMP) [20]. The matrix profile only stores information of the 'smallest distances'. It does not directly locate the nearest non-trivial neighbors. In order to fix this issue, a companion vector called matrix profile index is introduced.

Definition 7. A matrix profile index $I^T \in \mathcal{R}^{n-m+1}$ of time series T is a vector that stores the index of the nearest non-trivial neighbor of each subsequence. Formally, $I^T = [I_1^T, I_2^T, \cdots, I_{n-m+1}^T]$, where $I_i^T = j$ if $d(T_{i,m}, T_{j,m}) = nn_{nt}(D_i^T)$.

With access to the matrix profile index, we are able to locate the nearest non-trivial neighbor of any subsequence in a single query. The STAMP algorithm [20] discussed below is designed to output both matrix profile and matrix profile index of given input time series.

B. STAMP

STAMP is an efficient matrix profile calculation algorithm developed by research teams from the University of California Riverside and the University of New Mexico [21]. It is fast and simple to use. We provide the basic procedures of STAMP in Algorithm 1 and refer the readers to [20] for implementation details. The MASS function in line 4 computes the distance profile for a given subsequence. It is the key procedure in STAMP and demands the most computational resources. The ElementWiseMin function updates P^T and I^T by comparing P^T with D_i^T elementwisely. After obtaining all the distance profiles of subsequences, the matrix profile (index) is simply an ordered combination of minimum elements (locations) taken from each distance profile. Note that the only parameter we need to set for the entire algorithm is m, making STAMP easy to use in practice.

[1]The z-score scaling of subsequences requires either $O(nm)$ time plus $O(nm)$ space overhead or $O(n^2m)$ time overhead for the brute force algorithm.

Algorithm 1: STAMP

1 $m \leftarrow$ predefined value by user, $n \leftarrow$ length of T;
2 $P^T \leftarrow [inf, \cdots, inf]$, $I^T \leftarrow [0, \cdots, 0]$;
3 **for** *each subsequence* $T_{i,m}$ *in* T **do**
4 $\quad D_i^T \leftarrow$ MASS$(T_{i,m}, T, m)$;
5 $\quad P^T, I^T \leftarrow$ ElementWiseMin(P^T, I^T, D_i^T, i);
6 return P^T, I^T

The fundamental advantage of STAMP relies on its intriguing usage of fast Fourier transform (FFT) in MASS, which leads to a time complexity of $O(n^2 \log(n))$ for the entire algorithm. Specific description, implementation details, and complexity analysis of MASS are presented in [22]. The core idea is to exploit the similarities between convolution operation and Euclidean distance calculation. Some may argue that the reduction of time complexity from $O(n^2m)$ to $O(n^2 \log(n))$ is inconsequential. For small time series and small m, it is true that the difference between STAMP and the brute force approach is insignificant. However, as m and n grow, STAMP performs far better than its brute force opponent. For example, if we are processing a time series with $n = 50,000$ and $m = 1,000$, we would expect STAMP to be 64 times faster even without considering the heavy overhead of the brute force approach. Moreover, experiments carried out in [20] suggest the empirical run time of STAMP is roughly $O(n^2)$ instead of $O(n^2 \log(n))$ due to high efficiency of the existing FFT algorithms.

C. Mechanism of Event Discovery and Labeling

Fundamentally, using the matrix profile to discover and label system events in PMU data exploits the similarities of signatures created by similar events (disturbances). In this study, a signature is defined as a subsequence of PMU measurement time series corresponding to a system event (disturbance). Our mechanism of anomaly detection is highly dependent on the following two facts:

- First, the same type of system events usually have similar signatures in the time series.
- Second, signatures caused by the same type of system events have significantly lower euclidean distances in the matrix profile.

The first fact is straightforward and serves as the cornerstone of shape-based data analyses for event clustering [19] and classification [23] in power systems. The second fact is not apparent and might be somehow counterintuitive. Without occurrence of events, small subsequences of the frequency, voltage magnitude, and current magnitude time series are relatively steady and usually dominated by the DC components. Thus, the euclidean distances between them would be very small in theory. However, the real-world PMU data are always noisy [24], making these subsequences significantly different in terms of distance after z-score scaling. The signatures created by the same type of system events usually contain much more unique non-DC components, hence they remain

close even after z-score scaling. Therefore, the corresponding points of system events in the matrix profile are, in general, significantly smaller than that of normal conditions. By exploiting this property, the events can be automatically detected by locating the lowest points on the matrix profiles. This will be illustrated explicitly in the case study section. Meanwhile, their closest neighbors can be found simultaneously through the matrix profile index. This information can be used to cluster the system events directly. More importantly, given the small number of labeled data, our framework is able to automatically label the new system events by assigning them to their nearest neighbors.

III. Case Study

In this section, we show how to use a matrix profile to discover and automatically label power system events in real-world PMU data. All numerical studies are carried out in Python environment on a Dell desktop with a CPU of Intel Xeon E3-1226 v3 @ 3.30GHz. We start by describing our PMU dataset in the first subsection. The procedures of event discovery and labeling are discussed in the second subsection. Due to space limitation, only two case studies of generator and line tripping events are presented. Note that our proposed framework is very general and can be applied to discover a wide range of disturbances and events. In the last subsection, we show that the computation of the matrix profile is fast, scalable, and can be implemented in an anytime and incremental fashion (STAMP*I* [20]).

A. Data Source and Preprocessing

Our real-world PMU data have a reporting frequency of 30Hz, which contain two generator tripping events and two line tripping events. The two generator tripping events occurred consecutively and were therefore stored in the same file. The two line tripping events were recorded in two separate files. To show the matrix profile's ability to discover and automatically label events, we need at least two events from the same class. Thus, for the line tripping case, we concatenate the two time series into a single time series. Two modifications on the original datasets are made during the concatenation:

- First, the joint gap is smoothed out by aligning the terminal point of the preceding time series with the initial point of the following time series.
- Second, we add an additional small white noise to the concatenated time series such that the distances between normal subsequences from both parts of the time series are at the same level.

Now, we have two files containing two different types of events. Both of them are ready to be used for testing the performance of matrix profile.

B. Event Discovery and Labeling Using Matrix Profile

We discover and label the power system events by applying matrix profile on the current magnitude and rate of change of frequency (ROCOF) time series from the PMU data files. The length of subsequence m in this study is set as 90

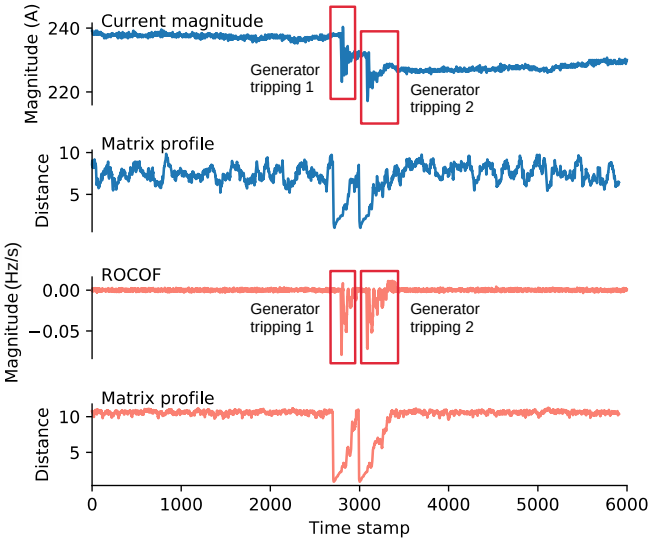

Fig. 2: Current magnitude and ROCOF time series from generator tripping event file and their matrix profiles.

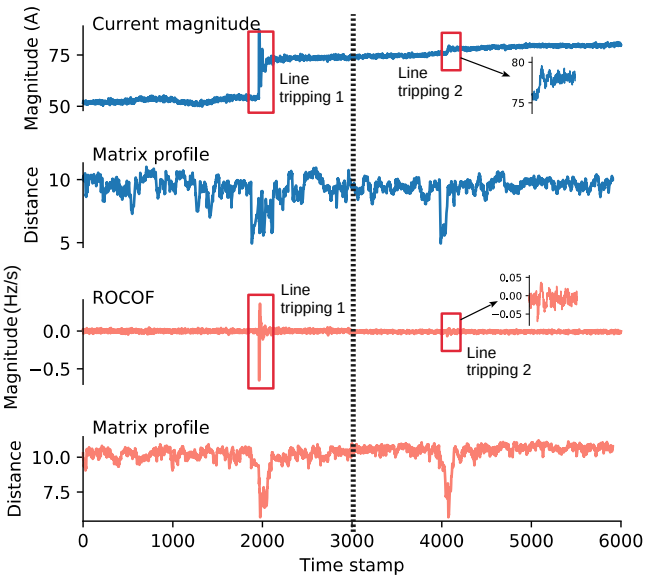

Fig. 3: Current magnitude and ROCOF time series from line tripping event file and their matrix profiles.

(3 seconds of PMU time series). Fig. 2 shows the current magnitude and ROCOF time series in the generator tripping event file and their corresponding matrix profiles. There are two outstanding 'valleys' in the matrix profiles that correspond to the two consecutive generator tripping events. To discover these events in power systems, we set a threshold $Th = \min(P^T) + [\max(P^T) - \min(P^T)]/4$ on the matrix profile. Then an event alert is sent whenever the distance in the matrix profile is below this threshold. Now suppose we do not know the label of the second event, then this event can be automatically labeled as a generator tripping event by finding its nearest neighbor through the matrix profile index.

Specifically, this process is implemented in three steps:

- First, we locate all the abnormal subsequences by finding local minima that are below the threshold in the matrix profile.
- Second, we find the nearest neighbor for each abnormal subsequence through the matrix profile index.
- Third, we assign the unlabeled subsequences with the same label as their nearest neighbors.

Unsurprisingly, the subsequences that correspond to the minima of the two 'valleys' in the matrix profile are nearest neighbors of one another. The second event is therefore labeled as a generator tripping event.

Fig. 3 depicts the current magnitude and ROCOF time series in the line tripping event file and their corresponding matrix profiles. The black dotted line marks the joint points of two separate time series. The position of the right hand side time series has been adjusted to smooth out the joint gap. Therefore, this figure does not reflect the real magnitude of the right hand side time series. Given the signature of the second line tripping event is small, this example also demonstrates the matrix profile's advantage in discovering minor power grid disturbances. Similar to the generator tripping case, the line tripping events can be alerted by setting a proper threshold on the matrix profile. The unlabeled events can then be automatically labeled in the same fashion.

C. Merits of a Matrix Profile Based Framework

In the introduction section, we argue that our framework is fast, scalable, and can be implemented in an anytime and incremental fashion. Now, we use numerical experiments to support these arguments. We will focus on the matrix profile computation process since it consumes most of the computational time. The running time of the remaining steps in the proposed event discovery and labeling framework is negligible compared with that of the matrix profile computation step.

1) Fast: Computation of a matrix profile is fast. To show this, we recalculate the matrix profile of the current magnitude time series in Fig. 2 (6,000 data points) for 50 times using Python codes. The time duration is, on average, 7.72 seconds per run. Note that this time can be even shorter if we use other programming languages such as C++ and Java.

2) Scalable: Computation of a matrix profile is scalable. Recall that the time complexity of STAMP is $\mathrm{O}(n^2 \log(n))$. It does not include the subsequence length m (the only user defined parameter). Thus, the time complexity of STAMP only depends on the length of a time series. To estimate the computation time of a matrix profile calculation for different lengths of the PMU data, we stitch multiple copies of the 200-second current magnitude time series together to construct longer time series. The computation time results are shown in Table I. Currently, we only use a single CPU core to run all the calculations. If we utilize multiple CPUs (or GPUs) and run the STAMP algorithm in a parallel fashion, the total computation time will be significantly reduced.

TABLE I: Computation time of a matrix profile (STAMP) and its update (STAMP*I*) for different lengths of PMU data

PMU data length	200 s	1 h	4 h	8 h
STAMP	7.72 s	36.97 min	13.75 h	57.86 h
STAMP*I*	0.0013 s	0.0205 s	0.1146 s	0.4822 s

Algorithm 2: STAMP*I*

1 $T^{new} = [T^{old}, d_{new}]$;
2 $n_{new} \leftarrow$ length of T^{new}, $s = n_{new} - m + 1$;
3 $D_{new} \leftarrow \mathrm{MASS}(T^{new}_{s,m}, T^{old}, m)$;
4 $P^T, I^T \leftarrow \mathrm{ElementWiseMin}(P^T, I^T, D_{new}, s)$;
5 $p_s, i_s \leftarrow \mathrm{Min}(D_{new})$;
6 $P^T \leftarrow [P^T, p_s]$, $I^T \leftarrow [I^T, i_s]$;
7 return P^T, I^T

3) Incremental Computability: The matrix profile can be updated incrementally. Note that the length of streaming synchrophasor time series grows quickly as time goes by. It is difficult and inefficient to update the matrix profile with the original STAMP algorithm. For example, if we have a PMU time series recorded for 8 hours ($\approx 2^{20}$ data points), it would take nearly 58 hours with our Dell desktop to calculate the new matrix profile when new data arrive. Although we can use multiple CPU cores to accelerate the computation speed, it is still too slow and inefficient. To overcome this difficulty, an incremental version of STAMP called STAMP*I* was proposed in [20]. The basic procedures of STAMP*I* are summarized in Algorithm 2. The Min function (line 5) returns both the minimum value p_s and its index i_s in D_{new}. By getting rid of the for-loop, the time complexity of STAMP*I* is $\mathrm{O}(n \log(n))$, that is n times faster than the original STAMP. To update the matrix profile of the same PMU time series recorded for 8 hours with one more measurement, the STAMP*I* algorithm only takes about 0.4822 seconds rather than 58 hours (see Table I). Therefore, STAMP*I* is highly recommended when updating the matrix profile of a large streaming dataset.

4) Anytime Algorithm: The matrix profile can be computed in an anytime fashion that gives decent approximate solution in a timely manner. Anytime algorithm refers to algorithms that can yield valid solutions to a problem even when they are interrupted in the middle of execution. A desirable anytime algorithm should obtain a decent approximate solution very fast and produce monotonically better results as the calculation process continues. The original STAMP algorithm can be adapted as an anytime algorithm by using a random sequence of subsequences in line 3 of Algorithm 1. The ElementWiseMin function guarantees that the solution is monotonically improving.

We test the anytime algorithm on the same current magnitude time series shown in Fig. 2. To measure the performance of STAMP*I*, we compute the mean absolute percentage error (MAPE) between the anytime solution and the final matrix profile as iteration continues. The results are depicted in Fig. 4. It shows that STAMP*I* can achieve a 90% accurate solution

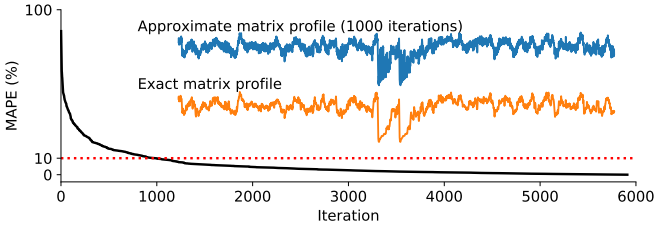

Fig. 4: Mean absolute percentage error between intermediate solution and final matrix profile as iteration continues.

after just 1,000 iterations (1/6 of the total iterations). This property can be very useful in online applications where the time window for calculation is very short. We do not always need the exact matrix profile. A decent approximate can work well for most power system event discovery and labeling.

IV. Conclusion

This paper proposes a novel data analytic framework with the matrix profile to discover and to automatically label power system events using weakly labeled data. The matrix profile can be efficiently computed through the STAMP algorithm. The distances in the matrix profile are relatively large during the normal operating conditions due to data noise, while the power system events from the same class usually have similar signatures in terms of shape. This property can be exploited to discover and label power system events in PMU data. Case studies with real-world PMU data are carried out to validate the proposed approach. The results show that our framework is capable of discovering and automatically labeling generator tripping and line tripping events. The numerical study also shows that the calculation of a matrix profile is fast, scalable, and can be implemented in an incremental and anytime fashion for online applications. Moreover, the anytime version of STAMP can provide a 90% accurate matrix profile using only 1/6 of the total required iterations.

In the future, we plan to investigate how to efficiently discover and label power system events with high-dimensional PMU data. Furthermore, we will explore whether the GPU-based matrix profile algorithms can better handle online power system event detection tasks. In addition, we are also interested in designing a systematic way of injecting noise into concatenated time series and an automatic threshold selection process for detecting anomalous subsequences in a matrix profile.

References

[1] S. Brahma, R. Kavasseri, H. Cao, N. Chaudhuri, T. Alexopoulos, and Y. Cui, "Real-time identification of dynamic events in power systems using PMU data, and potential applications-models, promises, and challenges," *IEEE Transactions on Power Delivery*, vol. 32, no. 1, pp. 294–301, Feb. 2017.

[2] A. Phadke and R. M. de Moraes, "The wide world of wide-area measurement," *IEEE Power and Energy Magazine*, vol. 6, no. 5, Sep. 2008.

[3] A. Phadke, J. Thorp, R. Nuqui, and M. Zhou, "Recent developments in state estimation with phasor measurements," in *Power Systems Conference and Exposition, PSCE'09*. IEEE, Mar. 2009, pp. 1–7.

[4] M. K. Neyestanaki and A. Ranjbar, "An adaptive PMU-based wide area backup protection scheme for power transmission lines," *IEEE Trans. Smart Grid*, vol. 6, no. 3, pp. 1550–1559, May 2015.

[5] H.-Y. Su, F.-M. Kang, and C.-W. Liu, "Transmission grid secondary voltage control method using PMU data," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 2908–2917, Jul. 2018.

[6] G. Anagnostou, F. Boem, S. Kuenzel, B. C. Pal, and T. Parisini, "Observer-based anomaly detection of synchronous generators for power systems monitoring," *IEEE Transactions on Power Systems*, vol. 33, no. 4, Jan. 2018.

[7] Y. Zhou, R. Arghandeh, H. Zou, and C. J. Spanos, "Nonparametric event detection in multiple time series for power distribution networks," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 2, pp. 1619–1628, Feb. 2019.

[8] A. Ukil and R. Živanović, "Abrupt change detection in power system fault analysis using adaptive whitening filter and wavelet transform," *Electric Power Systems Research*, vol. 76, no. 9-10, pp. 815–823, 2006.

[9] S. S. Negi, N. Kishor, K. Uhlen, and R. Negi, "Event detection and its signal characterization in PMU data stream," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3108–3118, Dec. 2017.

[10] D.-I. Kim, T. Y. Chun, S.-H. Yoon, G. Lee, and Y.-J. Shin, "Wavelet-based event detection method using PMU data," *IEEE Transactions on Smart grid*, vol. 8, no. 3, pp. 1154–1162, May 2017.

[11] M. He, J. Zhang, and V. Vittal, "Robust online dynamic security assessment using adaptive ensemble decision-tree learning," *IEEE Transactions on Power systems*, vol. 28, no. 4, pp. 4089–4098, Nov. 2013.

[12] M. Al Karim, M. Chenine, K. Zhu, L. Nordstrom, and L. Nordström, "Synchrophasor-based data mining for power system fault analysis," in *IEEE 3rd PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. IEEE, Oct. 2012, pp. 1–8.

[13] M. Biswal, Y. Hao, P. Chen, S. Brahma, H. Cao, and P. De Leon, "Signal features for classification of power system disturbances using PMU data," in *Power Systems Computation Conference (PSCC), 2016*. IEEE, Jun. 2016, pp. 1–7.

[14] M. Biswal, S. M. Brahma, and H. Cao, "Supervisory protection and automated event diagnosis using PMU data," *IEEE Transactions on Power Delivery*, vol. 31, no. 4, pp. 1855–1863, Aug. 2016.

[15] T. Yin, S. S. Wulff, J. W. Pierre, and T. J. Robinson, "A case study on the use of data mining for detecting and classifying abnormal power system modal behaviors," *Quality Engineering*, pp. 1–20, Jan. 2019.

[16] J. Ma, Y. V. Makarov, C. H. Miller, and T. B. Nguyen, "Use multi-dimensional ellipsoid to monitor dynamic behavior of power systems based on PMU measurement," in *Power and Energy Society General Meeting-conversion and Delivery of Electrical Energy in the 21st Century*. IEEE, Jul. 2008, pp. 1–8.

[17] L. Xie, Y. Chen, and P. R. Kumar, "Dimensionality reduction of synchrophasor data for early event detection: Linearized analysis," *IEEE Transactions on Power Systems*, vol. 29, no. 6, pp. 2784–2794, Nov. 2014.

[18] M. Zhou, Y. Wang, A. Srivastava, Y. Wu, and P. Banerjee, "Ensemble based algorithm for synchrophasor data anomaly detection," *IEEE Transactions on Smart Grid*, Mar. 2018.

[19] J. Cordova, C. Soto, M. Gilanifar, Y. Zhou, A. Srivastava, and R. Arghandeh, "Shape preserving incremental learning for power systems fault detection," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 85–90, Jan. 2019.

[20] 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 *IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, Dec. 2016, pp. 1317–1322.

[21] "The ucr matrix profile page," 2019, https://www.cs.ucr.edu/ eamonn/MatrixProfile.html.

[22] A. Mueen, Y. Zhu, M. Yeh, K. Kamgar, K. Viswanathan, C. Gupta, and E. Keogh, "The fastest similarity search algorithm for time series subsequences under euclidean distance," Aug. 2017, http://www.cs.unm.edu/ mueen/FastestSimilaritySearch.html.

[23] J. Cordova, R. Arghandeh, Y. Zhou, S. Wesolowski, W. Wu, and S. Matthias, "Shape-based data analysis for event classification in power systems," in *IEEE Manchester PowerTech*. IEEE, Jun. 2017, pp. 1–6.

[24] M. Brown, M. Biswal, S. Brahma, S. J. Ranade, and H. Cao, "Characterizing and quantifying noise in PMU data," in *IEEE Power and Energy Society General Meeting (PESGM)*. IEEE, Jul. 2016, pp. 1–5.