2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)

O-LoMST: An Online Anomaly Detection Approach And Its Application In A Hydropower Generation Plant

Imtiaz Ahmed[1], Travis Galoppo[2] and Yu Ding[3]

***Abstract*— With the increasing availability of streaming data, industries nowadays are striving for an automated online anomaly detection algorithm that can analyze data stream and detect anomalous patterns in real time. Such an online algorithm should detect anomalies on the fly, without storing all, or a very long stretch of, the historical data. It should be able to update its control mechanism for anomaly detection upon receiving new data. Moreover, the algorithm must work in an unsupervised way; i.e., in the absence of class labeling information *a priori*. These fundamental requirements limit the application of traditional anomaly detection approaches in streaming scenarios. In this paper, we introduce an online anomaly detection method, based on an offline method recently developed. The prototypical offline method is one of the new approaches that specifically handle the issue of nonlinear manifold embedding in data spaces and use a minimum spanning tree to approximate and capture the manifold structures, leading to a much enhanced detection ability. The primary objective of this paper is to make the offline method applicable to streaming data and address the aforementioned unique online issues. We elaborate the steps of our proposed approach by applying it to a hydropower generation plant and demonstrating how it can contribute to automation in that context.**

I. INTRODUCTION

Anomalies are data points or a cluster of data points which behave differently than their neighboring points or clusters and their characteristics do not match with the expected data distribution that represents the majority. Anomaly detection can be tricky, as anomalies can be global, segregable from the majority of data points, or they can be local, thus making it harder to isolate them unless compared with an appropriate group of neighboring points. It could be the abnormal value of a single attribute or a combination of values from multiple attributes that warn us about the presence of anomalies. Oftentimes, we do not have any prior information about the expected data distribution and we need to depend only on the structure of the dataset to isolate anomalies.

Detecting anomalies using unlabeled data streams pushes us into the territory of unsupervised learning and doing so is not an easy task. Consider the example of a hydropower plant, the real life process having motivated our work in the first place. It is operated with turbine systems that are instrumented with dozens of sensors. Each turbine has functional areas such as several bearing systems, a generator, etc. Sensors collect various types of data in real time such as temperature of oil inside the bearing systems, vibrations in each functional areas and many more. In total, each turbine collects more than 200 attributes from its sensors. This high dimensional data stream is then stored in a central control system and is available for evaluation in near real time. To protect the health of components it is vital to identify anomalies as they appear. But considering the number of attributes, identifying anomalies using visual plotting is not practical. Furthermore, as data streams arrive in real time, examining them one by one manually is not a feasible approach either. Unsurprisingly, a pressing need is to have an automatic *online* detection technique that can flag the appearance of potential anomalies as data is arriving.

The technical challenges of an online anomaly detection approach come in two folds. The high dimensional nature of the data stream poses the first set of challenges, because using Euclidean distances, which is the most widely used distance metric in existing anomaly detection algorithms, does not perform well in those cases [1]. Due to the noise aggregation in high dimensional data spaces, pairwise Euclidean distances become less differentiating for similar cases, and as a result, the Euclidean distance based discriminative methods may not be able to distinguish among the relative position of data points accurately. The higher dimensional space is more likely to embed a complicated structure thus leading to a nonlinear manifold. In the presence of such complicated space structure, the direct use of Euclidean distance between two points does not represent their intrinsic distance; please refer to the illustrative example in [2, Fig. 3]. The current solution is to use a geodesic distance instead of Euclidean distance in the presence of manifold structure to reflect more accurately the distance between the data points. The tricky part is that it is not straightforward to measure the geodesic distance when one does not know the underlying structure.

This first challenge has been more or less addressed in the form of an offline anomaly detection method in our recent publication [3]. By "offline," we mean that the historical data are available in either its entirety or a large sufficiency to the analysis algorithm, and that the analysis algorithm conducts one-time retrospective analysis. The solution approach in [3] is to devise a new similarity measure based on the concept of minimum spanning tree (MST). MST [4], [5] has the capability of approximating the geodesic distance in the presence of an embedded nonlinear manifold using the knowledge of neighboring points only. To implement the said similarity measure, one needs to convert the data points into a connected dense graph where each node represents an original data point and the edges between any two nodes represent the Euclidean distance between them. Then

[1]Imtiaz Ahmed with the Department of Industrial & Systems Engineering, Texas A&M University, College Station, TX imtiazavi@tamu.edu

[2]Travis Galoppo with ABB Corporate Research, Raleigh, NC travis.galoppo@us.abb.com

[3]Yu Ding with the Department of Industrial & Systems Engineering, Texas A&M University, College Station, TX yuding@tamu.edu

978-1-7281-0356-3/19/$31.00 ©2019 IEEE

individual local MSTs are built for each node in the graph using its neighboring points. The approach in [3] is based on the local MST distance measures and is referred to as LoMST. Empirical testing shows that LoMST outperforms a set of 13 competitive methods on 20 testing datasets of diversified sources.

Having (partially) addressed the first challenge, our attention now turns to the second set of challenges, arising from the streaming nature of the data. Traditional anomaly detection algorithms detect anomalies through the comparison of candidate data points with all observations in the historical data set. However, the same cannot be done when data are arriving in the streaming form, i.e., the data appear as a single stream or in small batches. In streaming data, an algorithm can only see the current batch and has limited information about the past, because the algorithm cannot, or does not want to, store all the previous data. Instead the algorithm only "memorizes" a handful of the most recent historical data, in order to avoid large storage requirements or for other practical considerations. The offline LoMST [3] indeed needs the knowledge of the nearest neighbors based on the whole dataset. To reach a verdict about a current data points, the offline method uses the information of all the past and current data points; all these actions need to be avoided, should the algorithm be made online compatible to handle streaming data.

Instead of storing all past data points, a streaming algorithm should presumably devise and make use of an indicator variable that can summarize the common characteristics of the past observations. The algorithm should embody a provision to continually update this indicator to reflect information received via the arrival of new data, as well as update the decision threshold that will be used for deciding in real time whether an observation is an anomaly or not. All these are precisely the goals of our research undertaking reported in this paper, namely that we propose an online version of the LoMST, referred to as online LoMST (O-LoMST), which has the aforementioned online capabilities.

The rest of the paper unfolds as follows: In Section II, we briefly summarize the related works in anomaly detection. Section III describes our proposed online anomaly detection approach in detail. In Section IV, we describe the hydropower plant and the data generation mechanism, and give a brief overview of the dataset. Section V presents the results of the proposed method applied to the hydropower dataset. Through a comparison with the offline LoMST, we demonstrate that the O-LoMST method accomplishes the online objective, without inadvertently sacrificing its detection capability. Finally, we conclude the paper in Section VI.

II. LITERATURE REVIEW

Anomaly detection is a broad field. A great number of works have already been done in this field. However, very few of them can analyze high dimensional data streams and thus cannot be applied in an online manner in their current forms. These offline anomaly detection algorithms can be categorized into four major groups, namely *distance and density based methods*, *clustering based methods*, *subspace methods* and *ensemble based methods*. Distance based methods considers a point as an anomaly if it lies too far away from majority of the data points or its neighbors [6]–[8]. Density based methods [9]–[11] consider the varying clustering tendency of data points and label a data point as an anomaly if the density around it is considerably lower than the density around its neighbors. The clustering based approaches [12] identify clusters at first and then label those points as anomalies which do not belong to the regular clusters or form a small and sparse cluster away from the regular clusters. Ensemble methods [13] combine multiple models of the same or different nature and then invoke a committee to reconcile the outcomes of the individual methods into a final detection.

All of these approaches use Euclidean distance to distinguish anomalies from normal data points and as a result, underperform when the data embeds a manifold structure. Researchers linked the substandard performance of these approaches to high dimensionality and propose that the investigation of anomalies should be carried out in relevant subspaces rather than the original space. Subspace methods [14], [15] undoubtedly made progress, but finding out the right subspaces to explore is still a difficult problem to solve.

A number of online anomaly detection algorithms have also been proposed recently. The first variety includes those which are partially online, [16], [17] meaning that they have an initial offline phase to learn about the regular (normal) clusters and their characteristics, and then detect anomalies online using the knowledge from the offline phase. The online part is self adaptive and the control mechanism (e.g., cluster center) gets periodically updated according to the changes observed in incoming data. The second variety includes the online version of the subspace algorithms [18], [19] which have been introduced to deal with the difficulties associated with high dimensional data stream. There are also online algorithms [20], [21] that use statistical tests to detect anomalies in real time. All these online methods have the same limitations as their offline counterparts, which is that they do not have the provision to deal with data with inherent nonlinear manifold structure. That is why we believe that converting the offline LoMST still warrants the research effort. The resulting online method is to be explained in the sequel.

III. O-LoMST: An online local MST based anomaly detection algorithm

Let us start off by recapping the concept of MST. Consider an undirected graph $G = (V, E)$, where V represents the collection of nodes and E represents the collection of edges connecting these nodes pairwise. For each edge $e \in E$, a weight is associated with it. A minimum spanning tree is a subset of the edges in E that connects all the nodes together, without forming a cycle and with the minimum possible total edge weight. For more clear understanding, consider the example in Fig. 1, left panel, where there are 5 nodes and 8 edges connecting them in total. Each of the edges

has a unique edge length associated with it. If we want to connect all the nodes using the given edges without forming a cycle, there could be many such combinations with only one having the minimum total edge length, which is shown in the right panel. The edges in a thick black line represent the selected 4 edges from the 8 total edges. The resulting graph consists of the black edges only in the right panel is the MST for the initial connected graph.

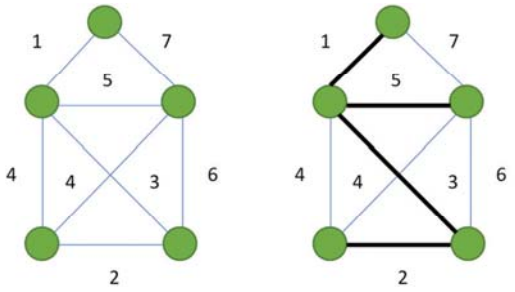

Fig. 1. Formation of a MST: the left panel is the initial graph, and the thick black edges form the minimum spanning tree in the right panel.

To construct the minimum spanning tree, at first we need to convert our attribute matrix into a graph object where each node represents a single data point and the connected edge between any two nodes represent the Euclidean distance between them. The MST is created out of this complete graph and is a sparse version of the original dense graph object. The distance between a pair of immediately connected nodes is still Euclidean, but the distance between a general pair of nodes is not. Rather, it is the summation of many small-step, localized Euclidean distances hopping from one data point to another, which converges to the geodesic distance as more data points are included.

To construct the regular, offline version of MST, one uses all the data points, which is not possible in the presence of the streaming data. In streaming scenarios, data points arrive either as a single entity or in small batches. Meanwhile, we do not want to store all the previous data points in memory while building and updating the MST, in order to be more efficient and save storage space.

To solve this problem, we use a local version of the regular MST in a streaming fashion. By "local MST," we mean that instead of using all the data points, the MST is to be constructed using the nearby points only. To facilitate the subsequent presentation, let us define some notations first. Denote each streaming batch by the index, j, and their size as B_j. Denote by t_j the data point that is currently under evaluation. To reduce the search space for neighbors, we choose to use a fixed-size candidate set of neighbors. It consists of the data points which are in close temporal vicinity to the time stamp under consideration, decided by an user controlled parameter, c. Given a c, the closest available c data points with respect to t_j form the candidate set. Certain care needs to be taken to locate the c closest neighbors. Although the neighbors are generally in the $\pm c/2$ range around t_j, a temporal truncation can make the data spread asymmetrically before and after t_j. For example, suppose that we are evaluating the 50^{th} time stamp from a batch that runs the time stamp index from 1 to 100 and $c = 30$. Then the time stamps ranging from 51–65 and 35–49 form the candidate set. But if the batch size is 60, meaning that the time stamp index runs from 1 to 60, then the time stamps 51–60 and 30–49 would form the candidate set instead.

Given a candidate set, Q_{t_j}, a neighborhood search is then conducted to find the k nearest neighbors of t_j, such that, $\|t_j - x_1\|_2 < \|t_j - x_2\|_2 < \cdots < \|t_j - x_k\|_2 < \cdots < \|t_j - x_c\|_2$, where $x_1, \ldots, x_k, \ldots, x_c \in Q_{t_j}$. We refer to the k nearest neighbors (NN), i.e., from x_1 to x_k, as the temporal k-NN of t_j and store them in $\mathfrak{N}_{t_j}$, i.e., $\mathfrak{N}_{t_j} = \{x_1, x_2, \ldots, x_k\}$.

After identifying the neighbors, we use those neighbor points along with t_j to build a dense graph object, i.e., fully connected graph. To create the MST from this dense graph, there are many different approaches. We choose to use Prim's algorithm [22] due to its effectiveness in the case of the fully connected graph. The algorithm selects a total of k edges from a collection of $\binom{k+1}{2}$ edges stored in $E_{t_j} = \{e_{t_j,x_1}, \cdots, e_{t_j,x_k}, e_{x_1,x_2}, \cdots, e_{x_1,x_k}, \cdots, e_{x_{k-1},x_k}\}$, such that sum of the length of the selected edges are minimum. Here e_{t_j,x_k} denotes the edge between two data points, t_j and x_k; other similar notations follow this convention. The selected edges by the Prim's algorithm are the edges in the resulting MST, connecting t_j and its neighbors. The total length of this MST is stored in W_{t_j}. This above step is then repeated for the remaining data points in batch j.

After a local MST is constructed for all time stamps in batch j, the connectedness of time stamp t_j is then compared with that of its neighbors, as expressed in (1), and the difference is denoted by $\mathfrak{L}_{t_j}$, such as

$$\mathfrak{L}_{t_j} = W_{t_j} - \overline{W}_{Nr_{t_j}}, \tag{1}$$

where $\overline{W}_{Nr_{t_j}}$ is the average of the total tree length associated with data points, other than t_j, in $\mathfrak{N}_{t_j}$. This comparison score, $\mathfrak{L}_{t_j}$, is to be used for the final anomaly evaluation.

The mean and standard deviation (SD) of all the comparison scores in a particular batch will be calculated, respectively, using (2) and (3), as:

$$\mu_j = \frac{1}{B_j} \sum_{t_j=1}^{B_j} \mathfrak{L}_{t_j}. \tag{2}$$

$$\sigma_j = \sqrt{\frac{1}{B_j} \sum_{t_j=1}^{B_j} (\mathfrak{L}_{t_j} - \mu_j)^2}. \tag{3}$$

These batch-specific mean and SD scores will be used to update the overall mean and SD from earlier batches, using the formula in (4) and (5), respectively, upon receiving new observations. Here μ_{old} and σ_{old} is the mean and SD value that are available to us prior to seeing the observations in the j-th batch.

$$\mu_{new} = \frac{1}{\sum_{i=1}^{j} B_i}\left[B_j \cdot \mu_j + \sum_{i=1}^{j-1} B_i \cdot \mu_{old}\right], \quad (4)$$

$$\sigma^2_{new} = \frac{1}{\sum_{i=1}^{j} B_i}\left[B_j \cdot \sigma^2_j + \sum_{i=1}^{j-1} B_i \cdot \sigma^2_{old}\right] + \frac{\sum_{i=1}^{j-1} B_i \cdot B_j}{(\sum_{i=1}^{j} B_i)^2} \cdot (\mu_{old} - \mu_j)^2. \quad (5)$$

To reflect any major change in the data generation process, (e.g., seasonal changes, maintenance operations, installation of new equipment etc.), we choose to divide the detection process into blocks and restart the mean and SD update process at the beginning of each block. The duration of each block can be selected by the domain expert based on the process under consideration. To decide on whether a time stamp is an anomaly or not, we use $\mu_{new} \pm 3\sigma_{new}$, the typical 3-sigma control limits, as the threshold. If $\mathfrak{L}_{t_j}$ is greater than the threshold, the corresponding t_j is marked as an anomaly. After a batch of streaming points are evaluated, the mean and SD scores will be updated and the same procedure will be repeated on the next available batch. Once all the batches from a block is completed, the detection process will restart again from the next block.

For a clear understanding, the algorithm steps are summarized in Algorithm 1.

Algorithm 1: O-LoMST algorithm for anomaly detection

Input : Block size, Z, streaming batch size, B_j in a block, number of candidate for neighbor selection, c, number of temporal neighbors, k

Output: List of anomalous time stamps from batch j, denoted as a_j

```
for each block of size Z do
    Initialize μ and σ;
    for each streaming batch j do
        for each time stamp t ∈ batch j do
            Form a set of candidate temporal neighbors, Q_{t_j};
            Determine the k nearest neighbors and save them in 𝔑_{t_j};
            Construct a local MST using time stamp t_j and its neighbors in 𝔑_{t_j};
            Calculate the total length of the resulting MST, W_{t_j};
            Calculate the mean, W̄_{𝔑_{t_j}}, of the total length of the local MSTs associated with all neighbors in 𝔑_{t_j};
            Calculate the LoMST score for t_j as 𝔏_{t_j} = W_{t_j} − W̄_{𝔑_{t_j}};
        end
        Calculate the mean, μ_j, and SD, σ_j, of the LoMST scores;
        Update the overall mean, μ, and SD, σ;
        List the time stamps as anomalies if 𝔏_{t_j} ≥ μ + 3σ and store them in a_j;
        Store the recent c/2 time stamps of batch j and discard others;
    end
end
```

IV. HYDROPOWER DATASET

To test our proposed streaming algorithm, we used a dataset generated from a hydropower plant. A typical hydropower plant is divided into several functional areas such as turbines, generators, bearing etc. Each area has many sub components and they are all equipped with smart sensors. These sensors across different functional areas continuously record the status of the machines and send them to the central control system. The time stamps are typically reported in 10 minutes of interval in each day. The common practice is that whenever a plant personnel suspects a possible malfunction, he/she extracts the data for a particular functional area and analyzes it manually to find the anomalies.

We received a total of seven months worth of data from five functional areas in a hydropower plant. First, we combine them into one .csv file for the ease of analysis. Initially there were 9,508 observations (rows in a data table) and 222 variables (columns in the data table). Variables include different temperatures, pressures, vibrations, power generated, ambient temperature etc. We did some basic data cleaning steps on the combined data file and find some obvious errors in the data collection process; please refer to our earlier report [23] for details. After getting rid of some of the time stamps in the preprocessing step, the number of observations came down to approximately 9,200. The initial analysis also gave us an indication about the clustering tendency in the data and the presence of local anomalies. The reduced dataset remains high dimensional and has 222 attributes.

V. EXPERIMENTAL SETTINGS & ANALYSIS OF RESULTS

In order to evaluate the performance of the proposed approach for streaming data, we pass the time stamps from the hydropower dataset in small batches to mimic the streaming scenario. We choose a fixed batch size, $B = 100$ and a block size, $Z = 7000$, meaning that in total there are 92 batches divided into 2 blocks to cover all of the time stamps. Selecting a suitable value for the neighborhood parameter, k, is not an easy task. This value is dependent on the size of the clusters present in the system. Ideally, we should select a value which is larger than a potential anomalous cluster but smaller than the regular cluster. After consulting with the our data provider and the domain expert, we have tried three different options, i.e., $k = 15, 20, 25$. For the temporal neighborhood size, c, we settle on the set of 50 candidate time stamps, i.e., $c = 50$, which is a half of the batch size.

After choosing these algorithmic parameters, we have applied our online detection method to all 92 batches and

detected a total number of 207 anomalous data points. The anomaly detection procedure regarding a particular batch is shown in Fig. 2. After each streaming batch, analysts can carefully evaluate the anomalies detected, pinpoint the root cause of these anomalies and take proper actions if necessary. Once we know the anomalies, we can follow up by employing a supervised classification technique that connects the anomalous outcomes with a specific combination of variable conditions and then establishing a set of rules for future anomaly detection; an example of such is given in [3]. We suggest the practitioners to stop the entire process if they detect too many anomalies, e.g., more than 60% in a batch. Because too many anomalies signal a major change in the process. Consequently operators should carefully examine the process for any major malfunction.

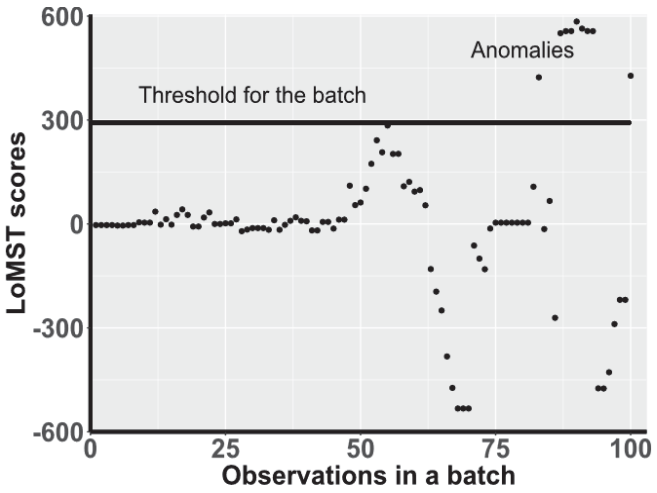

Fig. 2. Anomaly detection in a streaming batch

To save space, we report the top 100 anomalies, according to their LoMST scores (the bigger the score, the more likely it is an anomalous). We choose the top 100 anomalies so that we can compare them with the results of other anomaly detection methods from our earlier works [3], [23], including the offline version of LoMST, local outlier factor (LOF) [9], and subspace Outlying Degree (SOD) [14]. The detection outcomes by using, respectively, the aforementioned four approaches are summarized in Table I. Due to the space limitation, we skip some rows in the table. We observe that findings from the online LoMST are very similar to those by the offline LoMST, which substantiates the success of the online LoMST approach. The findings of other three approaches are taken from our earlier work [3]. As this is an unsupervised problem, we do not have the luxury to have a training set containing a mix of normal and anomalous values. The only way to verify the sanity of the outcome is to ask the domain expert and the operator of the system to double check. They can analyze the detected anomalies one by one, inspect the particular components in the plant, and feedback to us with their understandings. We consider ourselves extremely lucky here as the domain experts confirm the reasonableness of the detection outcomes, which was also so reported in our earlier work.

TABLE I
SUMMARY OF THE TOP 100 ANOMALIES RETURNED BY THE FOUR APPROACHES.

O-LoMST	LoMST	LOF	SOD
4/16/2015 23:10	4/16/2015 16:00	4/16/2015 16:00	7/4/2015 0:00
7/4/2015 4:20	4/16/2015 23:10	4/16/2015 23:10	7/4/2015 4:20
7/4/2015 4:30	7/4/2015 4:40	9/13/2015 7:00	7/4/2015 4:30
7/4/2015 4:40	7/4/2015 4:50	9/13/2015 19:30	7/4/2015 4:40
9/13/2015 19:00	7/4/2015 5:00	9/13/2015 19:40	7/4/2015 5:30
9/13/2015 21:40	7/4/2015 5:20	9/14/2015 0:40	7/4/2015 5:40
9/14/2015 0:40	7/4/2015 5:30	9/14/2015 1:00	7/4/2015 5:50
9/14/2015 1:00	7/4/2015 6:20	9/14/2015 1:10	7/4/2015 6:20
9/14/2015 1:10	7/4/2015 9:10	9/14/2015 2:00	7/4/2015 6:30
9/14/2015 1:50	7/4/2015 9:30	9/14/2015 2:10	7/4/2015 6:50
9/14/2015 8:00	7/4/2015 9:40	9/14/2015 8:00	7/4/2015 7:00
9/14/2015 8:10	7/4/2015 9:50	9/14/2015 8:10	7/4/2015 7:50
9/14/2015 8:20	7/4/2015 10:10	9/14/2015 8:20	7/4/2015 8:20
9/14/2015 8:30	7/4/2015 10:20	9/14/2015 8:30	7/4/2015 8:30
9/14/2015 8:40	7/4/2015 10:30	9/14/2015 8:40	9/13/2015 7:00
9/14/2015 8:50	7/4/2015 10:40	9/16/2015 10:50	9/13/2015 21:40
9/14/2015 9:00	7/4/2015 10:50	9/17/2015 11:30	9/14/2015 1:00
9/14/2015 9:10	7/4/2015 11:10	10/3/2015 14:40	9/14/2015 1:10
9/14/2015 9:20	7/4/2015 11:20	10/4/2015 3:10	9/14/2015 1:50
.....	.....	.....	.....
9/15/2015 20:50	7/4/2015 13:50	10/13/2015 5:45	9/14/2015 8:00
9/16/2015 10:30	9/13/2015 7:00	10/13/2015 6:35	9/14/2015 8:10
9/16/2015 10:50	9/13/2015 19:10	10/13/2015 8:15	9/14/2015 13:05
9/16/2015 11:00	9/14/2015 1:00	10/14/2015 7:55	9/16/2015 10:50
9/17/2015 11:30	9/14/2015 1:10	10/14/2015 8:15	9/16/2015 11:00
10/3/2015 14:40	9/14/2015 8:00	10/14/2015 23:15	10/3/2015 14:40
10/4/2015 3:10	9/14/2015 8:10	10/14/2015 23:35	10/4/2015 3:10
10/4/2015 3:40	9/14/2015 13:20	11/2/2015 9:56	10/4/2015 4:20
10/4/2015 4:10	9/14/2015 13:50	1/2/2016 9:10	10/4/2015 4:30
10/4/2015 4:20	9/14/2015 14:10	1/2/2016 9:20	10/13/2015 5:45
10/4/2015 4:30	9/16/2015 10:50	1/2/2016 9:30	10/13/2015 6:35
10/4/2015 9:00	10/13/2015 8:15	1/2/2016 21:40	10/13/2015 8:25
.....	.....	.....	.....
10/13/2015 5:25	10/14/2015 7:25	1/9/2016 6:50	10/14/2015 7:25
10/13/2015 5:35	10/14/2015 7:35	1/9/2016 18:00	11/2/2015 9:56
10/13/2015 5:45	1/2/2016 9:10	1/9/2016 18:10	1/2/2016 1:30
10/14/2015 7:25	1/2/2016 9:20	1/9/2016 18:20	1/2/2016 9:10
10/14/2015 7:35	1/2/2016 9:30	1/9/2016 18:30	1/2/2016 9:20
10/14/2015 7:55	1/9/2016 6:50	1/9/2016 18:40	1/2/2016 21:40
.....	.....	.....	.....
1/9/2016 18:00	1/9/2016 18:40	1/11/2016 11:30	1/9/2016 6:50
1/9/2016 18:10	1/11/2016 1:30	1/11/2016 11:40	1/9/2016 18:30
1/9/2016 18:20	1/11/2016 11:50	1/11/2016 11:50	1/11/2016 1:30
1/9/2016 18:30	1/11/2016 12:00	1/11/2016 12:00	1/11/2016 11:30
1/9/2016 18:40	1/11/2016 13:00	1/11/2016 13:00	1/11/2016 12:00
1/11/2016 13:40	1/11/2016 13:50	1/11/2016 13:50	1/11/2016 13:50
1/11/2016 13:50	1/12/2016 11:20	1/11/2016 14:40	1/11/2016 14:40
1/11/2016 14:40	1/12/2016 11:30	1/12/2016 11:20	1/12/2016 11:20
1/12/2016 11:20	1/12/2016 11:40	1/12/2016 11:30	1/12/2016 11:30
.....	.....	.....	.....

If we look very closely to the anomalous time stamps, we observe that anomalies from all four approaches have similar patterns and they can be conveniently grouped into some particular days. Even if the individual time stamps are not exactly the same, they fall into the same group. We have listed these anomaly prone days in Table II, and we note that they are very similar irrespective of the detection methods used. If we compare the performance of online and regular LoMST approach, we can observe that, out of 11 anomaly prone days, they share 9 of them. Despite the online features employed by O-LoMST, for instance, using only the current batch combined with summary statistics from prior batches, the online algorithm does not seems to sacrifice the detection performance too much. Comparing LoMST with LOF and SOD, the online LoMST in fact outperforms offline LOF

TABLE II

MOST ANOMALY PRONE DAYS ACCORDING TO THE FOUR METHODS

O-LoMST	LoMST	LOF	SOD
April 16th	April 16th	April 16th	July 4th
July 4th	July 4th	September 13th	September 13th
September 13th	September 13th	September 14th	September 14th
September 14th	September 14th	October 3rd	October 3rd
October 3rd	October 4th	October 4th	October 4th
October 4th	October 13th	October 13th	October 13th
October 13th	October 14th	October 14th	October 14th
October 14th	January 2nd	January 2nd	January 2nd
November 1st	January 9th	January 9th	January 9th
January 9th	January 11th	January 11th	January 11th
January 11th	January 12th	January 12th	January 12th

and SOD in terms of overall detection performance.

Finally, we would like to note that in the above comparisons, we only report the results under the parameter choice of $k = 15$. We indeed generated results using $k = 20$ and $k = 25$ but have not found any significant differences in the results. To save space, we omit the presentation of results under the other two choices of k.

VI. SUMMARY

The massive amount of data generated nowadays from the interconnected network of sensors and machines calls for real time processing due to storage limitations as well as the need to take immediate actions when warranted. Anomaly detection plays a vital role in many industries as accurate and timely detection of anomalies can save us from potential losses. Traditional anomaly detection algorithms are predominantly offline and require an online conversion to meet the real-time decision making needs. Our proposed online anomaly detection method is based on an offline MST-based detection method. But the online version entertains a number of advantages, such as it uses only the current batch of small number of observations, it summarizes information into the mean and standard deviation scores, it has a provision to continually update the mean and standard deviation scores to keep up with the underlying process, and it updates the decision threshold, upon receiving the new batch of observations, for deciding in real time whether an observation is an anomaly or not. When the online detection method is applied to a set of seven months of data from a hydropower plant, it detected 9 out of 11 failure prone days that were detected by using the offline detection method. This demonstrates the feasibility and effectiveness of online detection for handling streaming data. We believe this effort is among the important first steps moving from offline computation and detection towards the online paradigm.

REFERENCES

[1] A. Zimek, E. Schubert, and H.-P. Kriegel, "A survey on unsupervised outlier detection in high-dimensional numerical data," *Statistical Analysis and Data Mining*, vol. 5, no. 5, pp. 363–387, 2012.

[2] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[3] I. Ahmed, A. Dagnino, and Y. Ding, "Unsupervised anomaly detection based on minimum spanning tree approximated distance measures and its application to hydropower turbines," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 654–667, 2019.

[4] J. Costa and A. O. Hero, "Manifold learning with geodesic minimal spanning trees," *Computing Research Repository*, vol. cs.CV/0307038, 2003.

[5] W.-C. Tu, S. He, Q. Yang, and S.-Y. Chien, "Real-time salient object detection with a minimum spanning tree," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2334–2342.

[6] E. M. Knox and R. T. Ng, "Algorithms for mining distance based outliers in large datasets," in *Proceedings of the 24th International Conference on Very Large Data Bases*, 1998, pp. 392–403.

[7] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, vol. 29, no. 2, 2000, pp. 427–438.

[8] F. Angiulli and C. Pizzuti, "Outlier mining in large high-dimensional data sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 2, pp. 203–215, 2005.

[9] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, vol. 29, no. 2, 2000, pp. 93–104.

[10] E. Schubert, A. Zimek, and H.-P. Kriegel, "Local outlier detection reconsidered: A generalized view on locality with applications to spatial, video, and network outlier detection," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 190–237, 2014.

[11] L. J. Latecki, A. Lazarevic, and D. Pokrajac, "Outlier detection with kernel density functions," in *Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition*. Springer, 2007, pp. 61–75.

[12] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[13] A. Zimek, R. J. Campello, and J. Sander, "Ensembles for unsupervised outlier detection: Challenges and research questions a position paper," *ACM SIGKDD Explorations Newsletter*, vol. 15, no. 1, pp. 11–22, 2014.

[14] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Outlier detection in axis-parallel subspaces of high dimensional data," in *Advances in Knowledge Discovery and Data Mining: Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2009, pp. 831–838.

[15] B. Van Stein, M. Van Leeuwen, and T. Bäck, "Local subspace-based outlier detection using global neighbourhoods," in *IEEE International Conference on Big Data*, 2016, pp. 1136–1142.

[16] W. Wang, T. Guyet, R. Quiniou, M.-O. Cordier, F. Masseglia, and X. Zhang, "Autonomic intrusion detection: Adaptively detecting anomalies over unlabeled audit data streams in computer networks," *Knowledge-Based Systems*, vol. 70, pp. 103–117, 2014.

[17] M. Chenaghlou, M. Moshtaghi, C. Leckie, and M. Salehi, "Online clustering for evolving data streams with online anomaly detection," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2018, pp. 508–521.

[18] Y.-J. Lee, Y.-R. Yeh, and Y.-C. F. Wang, "Anomaly detection via online oversampling principal component analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 7, pp. 1460–1470, 2013.

[19] L. Akoglu and C. Faloutsos, "Event detection in time series of mobile communication graphs," in *Army Science Conference*, 2010, pp. 77–79.

[20] M. Solaimani, M. Iftekhar, L. Khan, and B. Thuraisingham, "Statistical technique for online anomaly detection using spark over heterogeneous data from multi-source vmware performance data," in *IEEE International Conference on Big Data*, 2014, pp. 1086–1094.

[21] A. Kejariwal, "Introducing practical and robust anomaly detection in a time series," *Twitter Engineering Blog*, 2015.

[22] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Labs Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

[23] I. Ahmed, A. Dagnino, A. Bongiovi, and Y. Ding, "Outlier detection for hydropower generation plant," in *Proceedings of the 14th IEEE International Conference on Automation Science and Engineering*, 2018, pp. 193–198.