Adaptive Cluster Tendency Visualization and Anomaly Detection for Streaming Data

DHEERAJ KUMAR, JAMES C. BEZDEK, SUTHARSHAN RAJASEGARAR, MARIMUTHU PALANISWAMI, CHRISTOPHER LECKIE, JEFFREY CHAN, and JAYAVARDHANA GUBBI, The University of Melbourne, Australia

The growth in pervasive network infrastructure called the Internet of Things (IoT) enables a wide range of physical objects and environments to be monitored in fine spatial and temporal detail. The detailed, dynamic data that are collected in large quantities from sensor devices provide the basis for a variety of applications. Automatic interpretation of these evolving large data is required for timely detection of interesting events. This article develops and exemplifies two new relatives of the visual assessment of tendency (VAT) and improved visual assessment of tendency (iVAT) models, which uses cluster heat maps to visualize structure in static datasets. One new model is initialized with a static VAT/iVAT image, and then incrementally (hence inc-VAT/inc-iVAT) updates the current minimal spanning tree (MST) used by VAT with an efficient edge insertion scheme. Similarly, dec-VAT/dec-iVAT efficiently removes a node from the current VAT MST. A sequence of inc-iVAT/dec-iVAT images can be used for (visual) anomaly detection in evolving data streams and for sliding window based cluster assessment for time series data. The method is illustrated with four real datasets (three of them being smart city IoT data). The evaluation demonstrates the algorithms' ability to successfully isolate anomalies and visualize changing cluster structure in the streaming data.

CCS Concepts: • **Information systems** → **Clustering and classification**; **Data stream mining**; **Online analytical processing**; *Hierarchical data models*; • **Computing methodologies** → **Anomaly detection**; • **Human-centered computing** → *Heat maps*; *Visual analytics*; • **Theory of computation** → *Streaming, sublinear and near linear time algorithms*; • **Mathematics of computing** → *Time series analysis*; Trees;

Additional Key Words and Phrases: Visual assessment of clusters in streaming data, cluster heat maps, internet of things (IoT), smart city streaming data analysis, online anomaly detection, sliding window based time series clustering

---

S. Rajasegarar, J. Chan, and J. Gubbi completed part of this work while at the University of Melbourne, Australia.
This work was supported by the Australian Research Council (ARC) Linkage Project Grant (LP120100529), the ARC Linkage Infrastructure, Equipment and Facilities scheme (LIEF) grant (LF120100129), EU-FP7 SOCIOTAL grant, and EU Horizon 2020 OrganiCity grant.
Authors' addresses: D. Kumar and M. Palaniswami, Department of Electrical and Electronic Engineering, University of Melbourne, Australia; emails: dheerajk@student.unimelb.edu.au, palani@unimelb.edu.au; S. Rajasegarar, School of Information Technology, Deakin University, Australia; email: sraja@unimelb.edu.au; J. C. Bezdek and C. Leckie, Department of Computing and Information Systems, University of Melbourne, Australia; emails: jcbezdek@gmail.com, caleckie@unimelb.edu.au; J. Chan, School of Science (Computer Science and Information Systems) RMIT University, Australia; email: jeffrey.chan@unimelb.edu.au; J. Gubbi, Embedded Systems and Robotics, TCS Research and Innovation, Bengaluru, India; email: jgl@unimelb.edu.au.

DOI: http://dx.doi.org/10.1145/2997656

ACM Reference Format:
Dheeraj Kumar, James C. Bezdek, Sutharshan Rajasegarar, Marimuthu Palaniswami, Christopher Leckie, Jeffrey Chan, and Jayavardhana Gubbi. 2016. Adaptive cluster tendency visualization and anomaly detection for streaming data. ACM Trans. Knowl. Discov. Data 11, 2, Article 24 (December 2016), 40 pages.
DOI: http://dx.doi.org/10.1145/2997656

1. INTRODUCTION

Data clustering is defined as the problem of dividing a set of n unlabeled objects $O = \{o_1, o_2, \ldots, o_n\}$ into k group of objects, where $k \leq n$. The partitioning is performed in such a way that objects belonging to the same cluster (group) are in some sense more similar to each other than to the objects belonging to different clusters. Most clustering approaches require the number of clusters to seek, k, as an input, but usually, this is not known in advance and needs to be estimated – this is the *cluster tendency* problem. Formally, data clustering is typically broken into a sequence of three computational steps:

—Assessment of cluster tendency – is there any cluster structure in the data? If yes, how many clusters (k) to seek?
—Partitioning the data into k clusters.
—Validation of the k clusters.

This article proposes a new visual assessment method for estimating the number of clusters when the data are evolving with time (streaming data). The approach depends on the creation of visual images that portray potential cluster structure using members of the *visual assessment of tendency* (VAT) and *improved* VAT (iVAT) models [Bezdek and Hathaway 2002; Hathaway et al. 2006; Havens and Bezdek 2012; Kumar et al. 2013, 2016]. VAT operates on an input dissimilarity matrix D, which can be obtained from input data vectors by computing the Euclidean distance between each pair of vectors. In a nutshell, VAT reorders the input dissimilarity data D to produce D^* using a modified version of the Prim's algorithm [Prim 1957] by connecting the next nearest datapoint to the current tree until the complete MST is formed. The *reordered dissimilarity image* (RDI) of D^*, $I(D^*)$, which, when displayed as a gray-scale image, shows possible clusters as dark blocks along the diagonal.

The widespread realization of the *Internet of Things* (IoT) produces an unprecedented amount of streaming data. Computationally efficient data analytic algorithms are needed to make sense of the voluminous data gathered. Cloud-centric infrastructure [Gubbi et al. 2013] supports the storage, analytics, and visualization of data. Nevertheless, extracting meaningful information from raw data is non-trivial. This encompasses both event detection and visualization of the associated raw and modeled data, with information represented according to the needs of the end-user. At present, there is no technique on offer for visualization of the point by point evolution of cluster structure in streaming data in real time. A trivial solution for the incremental problem will be to retrain VAT/iVAT each time a new datapoint arrives or is removed. However, this is not practical due to the computational complexities associated with the full retraining of the algorithm at each instance of the data arrival. In this article, we propose a "hot" update approach that can efficiently update the clustering mechanism. In particular, we propose a novel MST-based incremental update mechanism to achieve computationally efficient hot updates. We call our algorithm an incremental version of VAT and iVAT, viz., *incremental VAT* (inc-VAT) and *incremental iVAT* (inc-iVAT). For sliding time window based clustering applications, where we keep on adding new datapoints and removing the old ones, we propose a decremental version of VAT and iVAT, viz., *decremental VAT* (dec-VAT) and *decremental iVAT* (dec-iVAT). Figure 1 shows

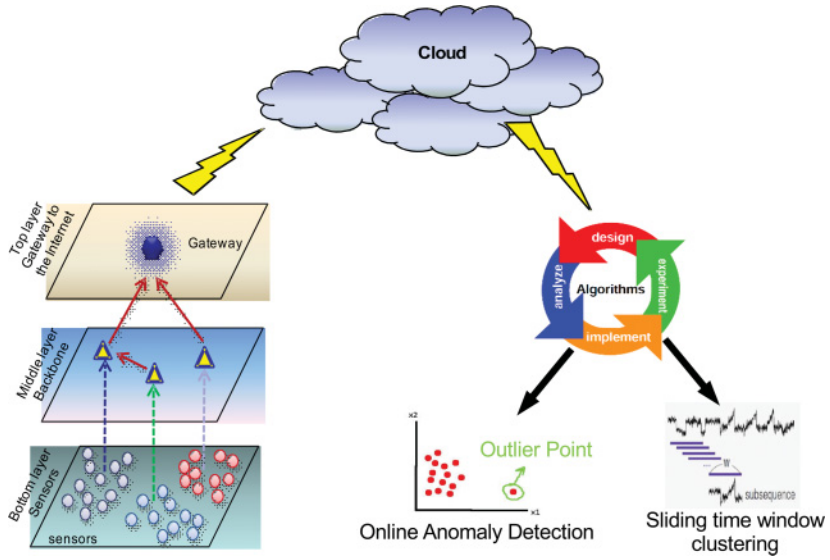


Fig. 1. Data collection and analysis system architecture.

an internal communication architecture regarding data flow for the end-to-end (from sensors to application) IoT realization.

Our main contributions in this article are as follows.

—We develop two new relatives of the VAT and iVAT model, which uses cluster heat maps to visualize structure in streaming datasets.
  —When a new datapoint arrives, inc-VAT/inc-iVAT incrementally updates the current minimal spanning tree (MST) used by VAT with an efficient edge insertion scheme.
  —dec-VAT/dec-iVAT efficiently removes a node from the current VAT MST.
—We perform experiments on a synthetic two-dimensional (2D) Gaussian mixture data to compare the CPU time of inc-VAT/dec-VAT and inc-iVAT/dec-iVAT to VAT/iVAT and the optimal dual-tree Boruvka algorithm for MST computation.
—We demonstrate the applicability of inc-iVAT/dec-iVAT for (visual) anomaly detection in evolving data streams and for sliding window based cluster assessment for time series data with four real-life datasets (three of them being smart city IoT data).

The rest of the article is organized as follows. Section 2 discusses related work. Section 3 covers the VAT and iVAT models and algorithms. Section 4 demonstrates the examples of anomaly detection and sliding window based clustering of time series data using inc-VAT and dec-VAT. Section 5 introduces the inc-VAT and inc-iVAT models, whereas the dec-VAT and dec-iVAT models are discussed in Section 6. The time complexity of these new algorithms is discussed in Section 7. Section 8 describes the use of inc-iVAT and dec-iVAT for finding clusters and anomalies, which is illustrated in Section 9 with three anomaly detection experiments and one sliding time window clustering experiment. Section 10 contains a discussion of the results, our conclusions, and lists some issues for further research.

2. RELATED WORK

Data clustering is primarily concerned with separating objects into k different groups. Let $O = \{o_1, o_2, \ldots, o_n\}$ denote n objects (guitars, shellfish, cricket fans, seismic

records, etc.). When each object in O is represented by a vector $\mathbf{x} \in \mathbb{R}^p$, the set $X = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^p$ is an object data or feature vector representation of O. The mth component of the ith feature vector ($\mathbf{x_{i_m}}$) is the value of the mth feature or attribute (e.g., height, weight, length, etc.) of the ith object o_i. Alternatively, when each pair of objects in O is represented by some relationship between them, we have relational data. The most common case of relational data is when we have (a matrix of) dissimilarity, say $D = [D_{ij}]$, where D_{ij} is the pairwise dissimilarity (usually a distance) $d(o_i, o_j)$ between objects o_i and o_j, for $1 \leq i, j \leq n$. We can convert X to D using a specified vector norm, so that $D = [d_{ij}] = [||\mathbf{x_i} - \mathbf{x_j}||]$.

Many clustering models are graph-theoretic, formally defined as follows. We define the undirected graph $G(\mathbb{V}, \mathbb{E})$, where the vertex set $\mathbb{V}$ represents the n objects in O as nodes so that $|\mathbb{V}| = n$. Since any node can be connected to any other node, we have $|\mathbb{E}| = \frac{n \times (n-1)}{2}$ edges and the weight given to the edge between o_i and o_j is D_{ij}. When D is a Euclidean distance matrix, a MST of $G(\mathbb{V}, \mathbb{E})$ is called a Euclidean MST (EMST). The simplest algorithm to find an EMST given the feature vector set $X \subset \mathbb{R}^p$, is to construct the complete graph $G(\mathbb{V}, \mathbb{E})$ on X, compute each edge weight by finding the Euclidean distance between each pair of feature vectors, and then run a standard minimum spanning tree algorithm (such as Prim's algorithm [Jarník 1930; Prim 1957; Dijkstra 1959] or Kruskal's algorithm [Kruskal 1956]) on it. Since the EMST has $O(n^2)$ edges for n distinct points, constructing it requires $O(n^2)$ time and space to store all the edges. There exist a large number of sophisticated general MST algorithms having different bounds on runtime. The current tightest bound of $O(\tau^*(|\mathbb{E}|, |\mathbb{V}|))$ was obtained in Pettie and Ramachandran [2002], where the function τ^* has a lower bound of $O(|\mathbb{E}|)$ and a upper bound of $O(|\mathbb{E}| \times \alpha(|\mathbb{E}|, |\mathbb{V}|))$, where α is the inverse Ackerman's function. However, all these general algorithms are insufficient for large, metric problems because they depend linearly on the number of edges. In the Euclidean case, the edge set consists of all pairs of points. Therefore, linear scaling in $|\mathbb{E}|$ corresponds to quadratic scaling in the number of points, n, and thus we need to consider other approaches. For 2D datapoints ($p = 2$), an EMST can be found in $O(|\mathbb{V}|\log(|\mathbb{V}|))$ time and $O(|\mathbb{V}|)$ space using Delaunay triangulation [Delaunay 1934]. Voronoi diagrams were used in Shamos and Hoey [1975] to construct an EMST in $O(|\mathbb{V}|\log(|\mathbb{V}|))$ for 2D and in $O(|\mathbb{V}|^2\log(|\mathbb{V}|))$ time for three or more dimensions. The tightest known lower bound of $O(|\mathbb{V}|\log(|\mathbb{V}|))$ for the EMST problem is given in Preparata and Shamos [1985]. A scalable dual-tree Boruvka algorithm for the EMST problem was presented in March et al. [2010]. For higher dimensions ($p \geq 3$), finding an optimal algorithm remains an open problem. In this article, we propose inc-VAT/inc-iVAT to update the VAT-generated EMST after new node insertion and dec-VAT/dec-iVAT for updating the EMST after node deletion.

Updating the MST after addition or deletion of a node is a fundamental problem for streaming data clustering. There are two main approaches for visualizing evolving cluster structure in streaming data besides cluster heat maps: cluster tracking and scatter plots.

Cluster tracking. Once clusters are obtained, the clusters can be visualized across time windows. Edges or lines are drawn between clusters that share membership and membership changes are shown by edges between different clusters in Greene et al. [2015]. This approach clearly shows the relative change in cluster memberships across time windows, but does not show the actual entities that experience membership changes. In Reda et al. [2011], the set of clusters at discrete time steps are drawn as spheres or represented by the thickness of lines. The focus of this type of dynamic visualization is to show (a) the relative size of the clusters and (b) how the cluster memberships change with time. In Angelov [2010] and Moshtaghi et al. [2015], chains of ellipsoids are used to depict evolving clusters and hyper-ellipsoidal boundaries provide

a mean for anomaly detection. All of these methods require found clusters as input, which in turn requires either advance knowledge or a method of "intelligent guessing" to specify the appropriate number of clusters to seek. The model parameter (k) is exactly what we hope to extract with inc-iVAT/dec-iVAT *prior* to clustering, and in fact, could be used in conjunction with any of these approaches. The XmdvTool project [XmdvTool 2015] provides a few density-based clustering approaches to streaming data [Yang et al. 2012, 2013a, 2013b], which are not directly comparable to the incremental hierarchical clustering approaches presented in this article.

Scatter plots. Another group of methods use scatterplots of the data across time [Elmqvist et al. 2008; Wong et al. 2003; Babcock et al. 2003; Lampe and Hauser 2011; Bezdek et al. 2007]. Scatter plots visualize data instances as points in 2D or 3D spaces. When visualizing data with a larger number of dimensions, typically the data is projected to a lower dimension, e.g., with multi-dimensional scaling [Wong et al. 2003] or *principle component analysis* (PCA). An alternative to multi-dimensional feature extraction is to project the data into many 2D scatterplots to try to get a sense of the relationship between the dimensions [Elmqvist et al. 2008]. These visualization approaches have also been extended to data streams. In data streams, either the method visualizes a sliding window of data [Babcock et al. 2003] or lines are drawn into kernelized density functions [Lampe and Hauser 2011]. However, these methods can fail when the clusters have high overlap or shapes that are difficult to distinguish on a scatter plot. When the number of instances is relatively high, particularly when we are visualizing many cases across a window of time, it can also be difficult to distinguish the clusters and points themselves, as well as the clustering trends (does this new cluster of points represents the movement of a subset of points in an older cluster?) If the data reside in more than three dimensions, this method engenders all of the difficulties inherent with feature extraction – what method of projection to use, loss of structural information, and so on.

In this article, we demonstrate two applications of visualizing evolving cluster structure in streaming data using inc-iVAT/dec-iVAT algorithms, viz. online anomaly detection and sliding window based clustering of time series data. Online anomaly detection is a significant problem in the IoT, where a sensor network monitors a phenomenon of interest over a geographical area. The events of interest are usually confined to a small duration of time (anomaly) in the otherwise huge amount of normal data. Previous works on anomaly detection include Sharma et al. [2010]'s rule-based and statistical modeling of anomalies, piecewise modeling of time series data [Yao et al. 2010], and change point detection based approaches for detecting distribution changes [Tartakovsky and Veeravalli 2008]. Rajasegarar et al. [2006, 2007] assumed the sensor system measurements to be from an unknown distribution and any measurement outside this distribution was considered as anomalous. In contrast, our work does not use any statistical assumptions about the input data, but uses VAT-based data clustering for automatically revealing the underlying structure and subsequently detecting anomalies.

Sliding time window based data clustering is a common approach for visualizing evolving structure in data for IoT smart city applications. The CluStream algorithm proposed in Aggarwal et al. [2003] clusters the data stream over different time horizons in a changing environment and stores the snapshots of clustering results over the landmark window, which consumes an enormous amount of space, and hence, is not suitable for the sliding window clustering problem. Zhou et al. [2008] combine the exponential histogram with temporal cluster features to define a novel data structure called *Exponential Histogram of Cluster Features* (EHCF). The exponential histogram is used to handle in-cluster evolution, and the temporal cluster features represent the

change of the cluster distribution. For this application, it is not sufficient to just add the latest datapoint to the VAT-generated MST using inc-VAT because of the memory requirement to accommodate the $O(n^2)$ size of the distance matrix, where n is the number of datapoints. A more appropriate strategy is to keep removing the oldest datapoints from the MST using dec-VAT to keep the number of datapoints manageable (assuming a sliding window of fixed size). Next, we describe the VAT and iVAT models regarding their algorithmic implementation.

3. VAT AND IVAT

The VAT algorithm reorders the input distance matrix D to obtain ordered dissimilarity matrix (D^*) using a modified Prim's algorithm that finds an MST of a weighted graph. Prim's algorithm proceeds by connecting the next nearest vertex to the current subtree until the complete MST is formed. VAT uses the order in which vertices (representing datapoints) are added to the MST and specifies a method for choosing the initial vertex for MST formation. It chooses either vertex forming the longest edge of the complete graph (whose edge weights are given by the input distance matrix). This choice initiates the ordering at one of the "outermost" points of the data. This initialization avoids the potential for zig-zag between possible clusters if the MST formation starts at an "interior" point.

The image $I(D^*)$ of the ordered dissimilarity matrix (D^*) may signal the presence of k well-separated clusters via the manifestation of k dark blocks of pixels on its main diagonal. This technique applies to any dissimilarity data that can be loaded into memory and all numerical data types. k clusters are then obtained by cutting the largest $k-1$ edges of the MST (given by the MST cut magnitude order d). Pseudocode for VAT is given in Algorithm 1.

Matrices are often used to describe k-partitions of X or D. The set M_{hkn}, given by Equation (1), is the set of all hard (non-soft) k-partitions of n objects.

$$M_{hkn} = \left[\begin{array}{ll} U \in \mathbb{R}^{kn} : u_{ij} \in \{0,1\} & \forall i,j \\ \sum_i u_{ij} = 1 & \forall j \\ 0 < \sum_j u_{ij} < n & \forall i \end{array} \right]. \tag{1}$$

The value of u_{ij} is 1 if object j is in cluster i; otherwise, $u_{ij} = 0$. All crisp (meaning neither fuzzy nor probabilistic) clustering algorithms generate solutions to the clustering problem for X or D by finding an "optimal" partition U in M_{hkn} that groups together subsets of vectors in X or objects in O. We presume that the objects in each cluster (identified by $1's$ in their rows in U) share some well-defined (mathematical) similarity. It is our hope, of course, that each mathematically optimal grouping is in some sense an accurate portrayal of natural groupings in the physical process from whence the object data are derived. But in this enterprise, there are no guarantees. Tendency assessment (and in particular, VAT and its relatives) attempts to find the best places to look for "good U's" – i.e., assessment of the data may suggest one, or at least a few, distinguished values of k to input to (any) clustering algorithm.

The set M^*_{hkn} of all aligned k-partitions is a subset of M_{hkn}. Aligned partitions have k continuous sub-blocks of adjacent $1s$ from left to right in the rows of U^*. For example,

$U = \left[\begin{array}{ccccc} 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{array} \right] \in M_{hkn}$ is unaligned; and

$U^* = \left[\begin{array}{ccccc} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{array} \right] \in M^*_{hkn}$ is aligned. We write

$U^* = \left[\begin{array}{ccccc} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{array} \right] \equiv \{1,1,1,2,2\} \equiv \{3:2\} \in M^*_{hkn}$.

ALGORITHM 1: VAT

Input : $D - n \times n$ dissimilarity matrix satisfying
- $D_{ij} \geq 0$
- $D_{ij} = D_{ji}\ \forall\ i, j$
- $D_{ii} = 0\ \forall\ i$

Output: $D^* - n \times n$ VAT reordered dissimilarity matrix
$P -$ VAT reordering indices of D^*
$d -$ MST cut magnitude order
$F -$ MST connection indices

Initialize the MST with the first element $\mathbf{x_1}$
$K = \{1, 2, ..., n\}$
$P_1 = 1$
$I = \{1\}$
$J = K \backslash \{1\}$
$F_1 = \{1\}$
Keep on adding the nearest of the remaining points to the current MST
for $t \leftarrow 2$ **to** n **do**
  $(i, j) = \arg \min_{m \in I, q \in J} D_{mq}$
  $d_{t-1} = D_{m_i q_j}$
  $P_t = J_j$
  $I = I \cup J_j$
  $J = J \backslash J_j$
  $F_t = i$
end
Rearrange the distance matrix D as per the VAT reordering indices P to obtain D^*
for $p \leftarrow 1$ **to** n **do**
  for $q \leftarrow 1$ **to** n **do**
    $D^*_{p,q} = D_{P_p, P_q}$
  end
end

This notation indicates there are three objects in the first cluster and two in the second. The order of the integers displays the order of the rows, left to right. An aligned k-partition that has n_i adjacent 1s in row i of U^* is completely specified by $\{n_1 : n_2 : \ldots : n_k\}$. Any *Single Linkage* (SL) partition U is an aligned partition U^* when U is reordered by VAT [Havens et al. 2009].

Although VAT can often provide a useful estimate of the number of clusters in a dataset, a much sharper RDI can usually be obtained using *improved VAT* (iVAT) as described in Wang et al. [2010] and Havens and Bezdek [2012]. iVAT provides better images by replacing input distances in the distance matrix $D = [d_{ij}]$ by geodesic distances $D' = [d'_{ij}]$, given by

$$d'_{ij} = \min_{p \in P_{ij}} \max_{1 \leq h \leq |p|} D_{p[h]p[h+1]}, \tag{2}$$

where P_{ij} is the set of all paths from object i (o_i) to object j (o_j) in the Euclidean graph of O. $p[h]$ is the index of the hth vertex along path p having a total of $|p|$ vertices, hence, $D_{p[h]p[h+1]}$ is the weight of the hth edge along path p. Essentially the cost of each path p, for the geodesic distance in Equation (2), is the maximum weight of its $|p|$ edges. We use the recursive version of iVAT presented in Havens and Bezdek [2012] as it has time complexity of $O(n^2)$ as compared to $O(n^3)$ for the iterative construction of D'^* in Wang

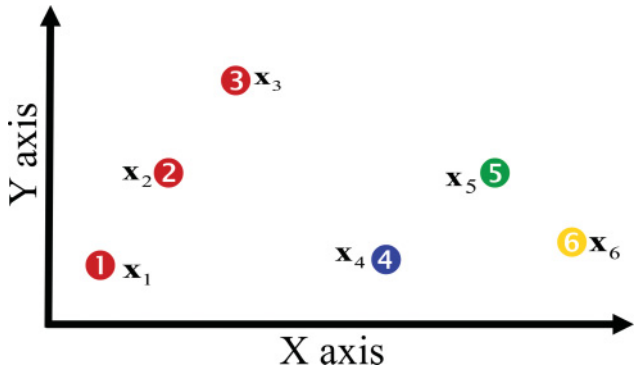


Fig. 2. Six 2D datapoints in a data stream.

et al. [2010]. Importantly, the theory that connects SL to VAT, also holds for recursive iVAT, which preserves the VAT reordering of D. Pseudocode for recursive iVAT is given in Algorithm 2.

ALGORITHM 2: iVAT

Input : $D^* - n \times n$ VAT reordered dissimilarity matrix
Output: $D'^* - n \times n$ iVAT dissimilarity matrix

for $r \leftarrow 2$ **to** n **do**
  $j = \arg \min_{1 \le m \le r-1} D^*_{rm}$
  $D'^*_{rj} = D^*_{rj}$
  $c = \{1, 2, \ldots, r-1\} \backslash \{j\}$
  $D'^*_{rc} = max\{D^*_{rj}, D'^*_{jc}\}$
  $D'^*_{rc} = D'^*_{cr}$
end

4. DEMONSTRATION EXAMPLE OF ANOMALY DETECTION AND SLIDING WINDOW BASED CLUSTERING OF TIME SERIES DATA

Before turning to the technical details, we illustrate the inc-VAT and dec-VAT models with a small (but actual) example. The basic idea is depicted in Figures 2, 3, and 4. Figure 2 shows a set of six points in $X = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_6}\} \subset \mathbb{R}^2$. We have numbered the points to correspond to six successive instants in time, such that $\mathbf{x_1}$ is the first and $\mathbf{x_6}$ is the last datapoint of the data stream. We convert X to a 6×6 dissimilarity matrix D, by computing the Euclidean distance between each pair of vectors in X, so that $D = [d_{ij}] = [||\mathbf{x_i} - \mathbf{x_j}||_2], 1 \le i, j \le 6$.

Figure 3(a) depicts a VAT reordered distance matrix D^*_3 of the first three rows and columns of D corresponding to the first three datapoints across time window TW_3 ending at time t_3. The three darker 1×1 blocks along the diagonal of the 3×3 VAT image in view (a) suggest that there may be three different singleton objects (we refrain from calling a single object a "cluster") underlying the data. However, the somewhat darker off-diagonal intensities create a 3×3 block capturing the darker diagonals, and this offers a different interpretation of structure in D, viz., that the three objects do share weak pairwise similarities to each other. The next sample, datapoint $\mathbf{x_4}$, arrives in Figure 3(b). Our new inc-VAT algorithm increments the 3×3 VAT reordered distance matrix D^*_3 by 1 row and 1 column (incrementally adding the fourth edge to the EMST of

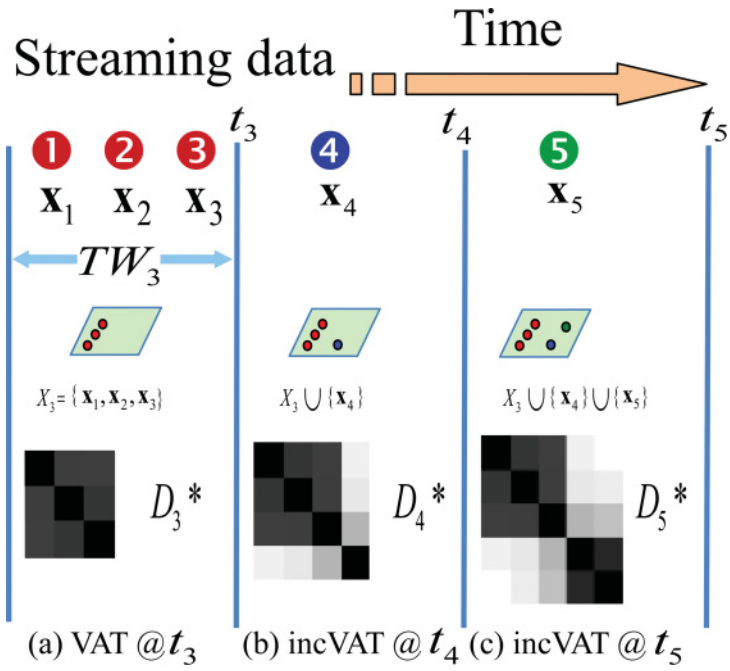


Fig. 3. An example illustrating the inc-VAT architecture.

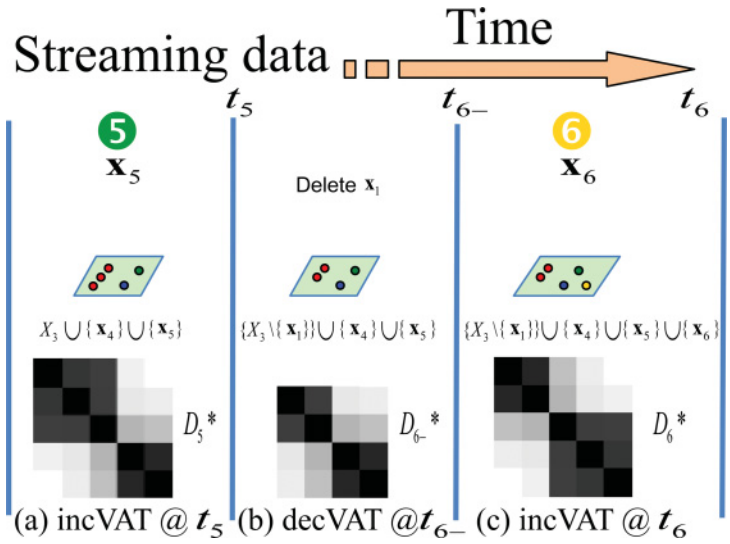


Fig. 4. An example illustrating the sliding window clustering application using dec-VAT and inc-VAT.

the current data), resulting in a 4×4 inc-VAT reordered distance matrix D_4^* shown in view (b). This view visually isolates the new point $\mathbf{x_4}$ by imaging it as a separate second dark diagonal block in the lower right corner of the image. Looking at the incoming data in Figure 2 confirms that this is a good visual cue of evolving structure in the data stream. In Figure 3(c), the fifth point, $\mathbf{x_5}$ arrives. inc-VAT updates the current inc-VAT reordered dissimilarity matrix D_4^*, MST, and RDI, resulting in the 5×5 image

D_5^*, which suggests that a second cluster containing two points has arisen in the data stream. Again, this seems like a good view of the evolving structure.

From this little example, we see that a typical application of this algorithm would be online anomaly detection, where we would expect a few samples that are quite different from others in the data stream to maintain separate pixels as the inc-VAT image evolves. This would be the case (temporarily here), for example, in Figure 3(b), where $\mathbf{x_4}$ establishes a new diagonal sub-bock that is "separate" from the main 3×3 block in Figure 3(a).

Another typical application of this algorithm would be sliding window based clustering of time series data. For this application, it is not sufficient to just add the latest datapoint to the VAT-generated MST using inc-VAT because of the memory requirement to accommodate the $O(n^2)$ size of the distance matrix. A more appropriate strategy is to keep removing the oldest datapoints from the MST using dec-VAT to keep the number of datapoints manageable (assuming a sliding window of fixed size). Figure 4 shows an example of our inc-VAT and dec-VAT algorithms for a sliding time window based clustering application. Figure 4(a) is the 5×5 inc-VAT image at time t_5 and let us assume we have chosen the window size to be 5. When the new datapoint $\mathbf{x_6}$ arrives at time t_6, dec-VAT deletes the oldest datapoint $\mathbf{x_1}$ from the inc-VAT image at t_5, generating a 4×4 dec-VAT image of $\{X_3 \setminus \{\mathbf{x_1}\}\} \cup \{\mathbf{x_4}\} \cup \{\mathbf{x_5}\}$, just before time t_6 (represented as t_{6-}) as shown in view (b). At time t_6, we add the new datapoint $\mathbf{x_6}$ to the MST using the inc-VAT algorithm providing the (updated) 5×5 inc-VAT image of $\{X_3 \setminus \{\mathbf{x_1}\}\} \cup \{\mathbf{x_4}\} \cup \{\mathbf{x_5}\} \cup \{\mathbf{x_6}\}$ as shown in view (c). The inc-VAT images in views (a) and (c) of Figure 4 clearly show the shrinking of cluster 1 (top left dark block) and swelling of cluster 2 (bottom right dark block) by one pixel each. This is supported by the streaming data structure, as $\mathbf{x_1}$ which was a part of cluster 1 has been removed and $\mathbf{x_6}$ which is a part of cluster 2 is added to the sliding window at t_6. Figures 3 and 4 illustrate the main objectives of this article: To develop and test the inc-VAT/dec-VAT (and inc-iVAT/dec-iVAT) models for visualization of evolving cluster structure in streaming data for online anomaly detection and sliding time window based clustering applications.

5. inc-VAT/inc-iVAT DESCRIPTION EXAMPLE AND ALGORITHM

The VAT algorithm works fine for small datasets ($n \leq 10{,}000$), but for larger values, platform constraints and the time complexity of $O(n^2)$ are limiting factors. For time series, where datapoints arrive sequentially, VAT needs to be (re)executed at every arrival, a time-consuming task. Consider a time series dataset $X_n = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^p$ having n p-dimensional datapoints arriving at time instants $\{1, 2, \ldots, n\}$. We consider $n \geq 2$ so that an initial VAT image of X_n can be constructed. When $\mathbf{x_{n+1}}$ arrives at time $n+1$, we get the augmented dataset $X_{n+1} = X_n \cup \{\mathbf{x_{n+1}}\}$. To find the new image, we would need to (re)compute the VAT algorithm for the entire dataset X_{n+1}, without reusing the results from the previous image of X_n. inc-VAT circumvents this problem with an incremental version of VAT that updates the current image by first computing n new distances between the incoming point $\mathbf{x_{n+1}}$ and the previous n points. Following this, one new edge is inserted into the current MST and the new image is displayed.

To assist with the following description of our algorithms, a summary of the notations used in Sections 5 and 6 is provided in Table I.

Let the dissimilarity matrix of X_n be D_n. If we apply VAT to D_n, we get an $n \times n$ reordered dissimilarity matrix D_n^*, VAT reordering indices P_n, the MST cut magnitude order d_n, and the MST connection indices F_n. VAT reorders the n datapoints (or objects) in such a way so that each point in the reordered matrix is closer than any datapoint after it to any datapoint before it (before and after here refer to positions in the reordered list of indices, not to times of data arrival). To

Table I. Symbols Used in Description of Algorithms

Symbol	Description
X_n	Time series dataset consisting of n p-dimensional datapoints $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$ arriving at time instants $\{1, 2, \ldots, n\}$
$\mathbf{x_i}$	ith p-dimensional datapoints arriving at time instant i
D_n	$n \times n$ dissimilarity matrix of X_n
D_n^*	VAT reorderd dissimilarity matrix of X_n
P_n	VAT reordering indices of X_n
d_n	MST cut magnitude order of X_n
F_n	Connection indices of the VAT-generated MST of X_n
$D_n^{'*}$	iVAT reorderd dissimilarity matrix of X_n
$V = \{v_1, v_2, \ldots, v_n\}$	Distance of the new datapoint $\mathbf{x_{n+1}}$ from $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$
$Y = V_{P_n}$	Distances of the new datapoint $\mathbf{x_{n+1}}$ from the VAT reordered datapoints of X_n
A	Set of indices in P_n, which are not in P_{n+1}
C	LongestSubsequence(P_n, P_{n+1})
Group 1 (G1)	Datapoints whose indices form the longest subsequence C of P_n, and whose elements are present in P_{n+1}
Group 2 (G2)	The new datapoint $\mathbf{x_{n+1}}$
Group 3 (G3)	Datapoints corresponding to the remaining indices in P_{n+1} which are not in G1 or G2
B	$P_n \backslash C$
H	MST connection indices of the datapoints in B
E	$P_{n+1} \backslash \{n+1\} \backslash C$
$w_i : 1 \leq i \leq 3$	Index of the datapoint represented by indices in A which is closest to G1, G2, and G3 for i = 1, 2, and 3, respectively
$z_i : 1 \leq i \leq 3$	Distance of w_1, w_2, and w_3 from the closest point in G1, G2, and G3 for i = 1, 2, and 3, respectively
$v_i : 1 \leq i \leq 3$	Positions of the datapoints in G1, G2, and G3, in P_{n+1}, which are closest to w_1, w_2, and w_3, respectively
G	Reordering indices of P_n to obtain P_{n+1}
J	The set of datapoints after position i in P_n that are connected to the to-be-removed point $\mathbf{x_p}$ in the current MST of X_n

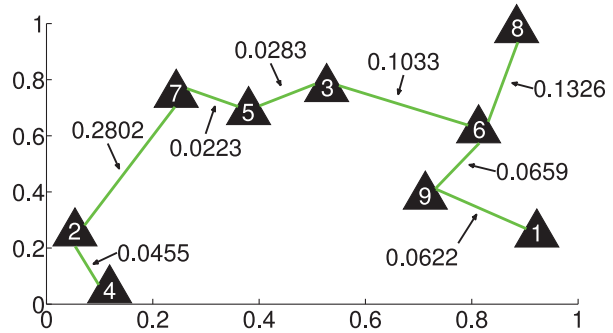


Fig. 5. MST for X_9.

illustrate this, consider the dataset X_9 of nine 2D datapoints shown in Figure 5, which are originally labeled as $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_9}\}$. When we apply VAT to X_9, the reordered array of indices is $P_9 = \{1, 9, 6, 3, 5, 7, 8, 2, 4\}$, the MST cut magnitude order is $d_9 = \{0.0622, 0.0659, 0.1033, 0.0283, 0.0223, 0.1326, 0.2802, 0.0455\}$, and the MST connection indices are $F_9 = \{1, 1, 2, 3, 4, 5, 3, 6, 7\}$. Figure 5 also shows the MST edges of X_9 and corresponding edge lengths represented by d_9. Hence, the VAT reordering of these nine points obtained using P_9 is $\{\mathbf{x_1}, \mathbf{x_9}, \mathbf{x_6}, \mathbf{x_3}, \mathbf{x_5}, \mathbf{x_7}, \mathbf{x_8}, \mathbf{x_2}, \mathbf{x_4}\}$, and we can

conclude that out of $\{\mathbf{x_9}, \mathbf{x_6}, \mathbf{x_3}, \mathbf{x_5}, \mathbf{x_7}, \mathbf{x_8}, \mathbf{x_2}, \mathbf{x_4}\}$, $\mathbf{x_9}$ is closest to $\mathbf{x_1}$. Similarly, out of $\{\mathbf{x_6}, \mathbf{x_3}, \mathbf{x_5}, \mathbf{x_7}, \mathbf{x_8}, \mathbf{x_2}, \mathbf{x_4}\}$, $\mathbf{x_6}$ is closest to any one of $\{\mathbf{x_1}, \mathbf{x_9}\}$, and so on.

This property of VAT reordering is the basis for inc-VAT, an incremental version of VAT. The inc-VAT algorithm comprises two main steps:

(1) Finding the insertion position of the new datapoint $\mathbf{x_{n+1}}$ in the VAT ordering of X_n.
(2) Reordering the remaining datapoints after the insertion position.

We now describe each of these steps in more detail and formulate the inc-VAT and inc-iVAT algorithms.

5.1. Step 1: Finding the Insertion Position of the New Datapoint

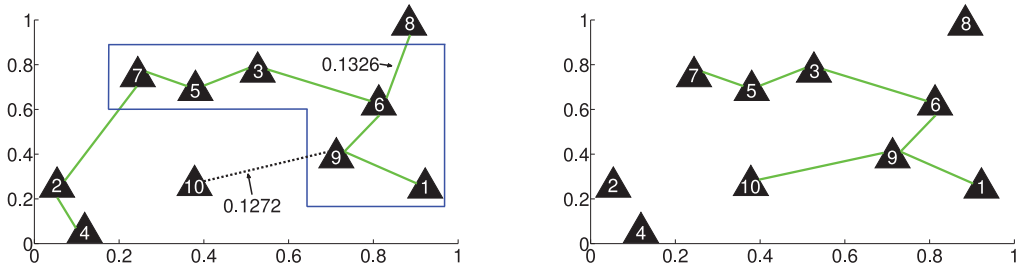
When the new datapoint $\mathbf{x_{n+1}}$ arrives, we compute the distances of $\mathbf{x_{n+1}}$ to all the points in X_n. Let these n new distances be denoted by $V = \{v_1, v_2, \ldots, v_n\}$, where, v_1 is the distance between $\mathbf{x_1}$ and $\mathbf{x_{n+1}}$, v_2 is the distance between $\mathbf{x_2}$ and $\mathbf{x_{n+1}}$, and so on. Rearranging V using the indices in P_n gives us Y, the distances of the new datapoint from the VAT reordered datapoints of X_n. $Y = V_{P_n} = \{v_{P_{n_1}}, v_{P_{n_2}}, \ldots, v_{P_{n_n}}\}$. The procedure to find the insertion position (say i) of the new datapoint in the VAT reordering of the augmented set X_{n+1} is given in Algorithm 3. The new index array for dataset X_{n+1}, P_{n+1} is initialized with the first $i-1$ elements of P_n and appends the new index $n+1$ at the end, hence we initialize $P_{n+1} = \{P_{n_1}, P_{n_2}, \ldots, P_{n_{i-1}}, n+1\}$. Similarly, the new MST cut magnitude order is initialized as $d_{n+1} = \{d_{n_1}, d_{n_2}, \ldots, d_{n_{i-2}}, \min\{Y_1, Y_2, \ldots, Y_{i-1}\}\}$ and the new MST connection indices are initialized as $F_{n+1} = \{F_{n_1}, F_{n_2}, \ldots, F_{n_{i-1}}, \text{argmin}(\{Y_1, Y_2, \ldots, Y_{i-1}\})\}$ as shown in Algorithm 3.

ALGORITHM 3: InsertPosition

Input : P_n − VAT reordering indices of D_n^*
d_n − MST cut magnitude order of D_n^*
F_n − MST connection indices of D_n^*
$V = \{v_1, v_2, \ldots, v_n\}$ − Distance of $\mathbf{x_{n+1}}$ from $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$
Output: i − Insertion position of $\mathbf{x_{n+1}}$
P_{n+1} − Initialization of VAT reordering indices of D_{n+1}^*
d_{n+1} − Initialization of MST cut magnitude order of D_{n+1}^*
F_{n+1} − Initialization of MST connection indices of D_{n+1}^*

```
Y = V_{P_n} = {v_{P_1}, v_{P_2}, ..., v_{P_n}}
i = n + 1
j = argmin(V)
for t ← 1 to n − 1 do
    if min({Y_1, Y_2, ..., Y_t}) < d_{n_t} then
        i = t + 1
        j = argmin({Y_1, Y_2, ..., Y_t})
        break
    end
end
P_{n+1} = {P_{n_1}, P_{n_2}, ..., P_{n_{i−1}}, n + 1}
d_{n+1} = {d_{n_1}, d_{n_2}, ..., d_{n_{i−2}}, min(Y_1, Y_2, ..., Y_{i−1})}
F_{n+1} = {F_{n_1}, F_{n_2}, ..., F_{n_{i−1}}, j}
```

In the insertion example (Figures 5, 6, and 7), when the new datapoint $\mathbf{x_{10}}$ arrives, we get the augmented dataset X_{10} as shown in Figure 6(a). Distances of the new datapoint $\mathbf{x_{10}}$ from the datapoints in X_9 is $V = \{0.2963, 0.1054, 0.2793, 0.1115,$



(a) New datapoint $\mathbf{x_{10}}$ (b) Insertion of $\mathbf{x_{10}}$ in MST of X_9

Fig. 6. Finding insertion position of the new datapoint $\mathbf{x_{10}}$.

Table II. LongestSubsequence Examples

P_n	P_{n+1}	LongestSubsequence(P_n, P_{n+1})
{1, 9, 6, 3, 5, 7, 8, 2, 4}	{1, 9, 6, 3, 5, 7, 10}	{1, 9, 6, 3, 5, 7}
{1, 9, 6, 3, 5, 7, 8, 2, 4}	{1, 9, 6, 3, 5, 7, 10, 2}	{1, 9, 6, 3, 5, 7}
{1, 9, 6, 3, 5, 7, 8, 2, 4}	{1, 9, 6, 3, 5, 7, 10, 2, 8}	{1, 9, 6, 3, 5, 7, 8, 2}
{1, 9, 6, 3, 5, 7, 8, 2, 4}	{1, 9, 6, 3, 5, 7, 10, 4, 2}	{1, 9, 6, 3, 5, 7}
{1, 9, 6, 3, 5, 7, 8, 2, 4}	{1, 9, 6, 3, 5, 7, 10, 4, 2, 8}	{1, 9, 6, 3, 5, 7, 8, 2, 4}

0.1807, 0.3186, 0.2546, 0.7705, 0.1272} and when reordered as per the VAT reordering P_9 of X_9, gives $Y = \{0.2963, 0.1272, 0.3186, 0.2793, 0.1807, 0.2546, 0.7705, 0.1054, 0.1115\}$. The condition for determining the position i of the new datapoint in the inc-VAT reordering of X_{10} is satisfied at $i = 7$ as $\min\{Y_1, Y_2, \ldots, Y_6\} = Y_2 = 0.1272$ is smaller than $d_{9_6} = 0.1326$. Hence, $\mathbf{x_{10}}$ is inserted at position 7 in the VAT reordering of X_9 as shown by the dotted black edge in Figure 6(a). The new index array for X_{10} is initialized to $P_{10} = \{1, 9, 6, 3, 5, 7, 10\}$, and similarly, $d_{10} = \{0.0622, 0.0659, 0.1033, 0.0283, 0.0223, 0.1272\}$ and $F_{10} = \{1, 1, 2, 3, 4, 5, 2\}$. This incomplete MST just after insertion of $\mathbf{x_{10}}$ is shown in Figure 6(b) with $\mathbf{x_{10}}$ connected to $\mathbf{x_9}$ and $\{\mathbf{x_2}, \mathbf{x_4}, \mathbf{x_8}\}$ disconnected from the MST.

5.2. Step 2: Reordering the Remaining Datapoints After the Insertion Position

Next, we reorder the remaining indices $\{P_{n_i}, P_{n_{i+1}}, \ldots, P_{n_n}\}$, where i is the insertion position of $\mathbf{x_{n+1}}$ using the procedure IncInsert (Algorithm 6). Let these remaining indices be represented by A, which is the set of indices in P_n, which are not in P_{n+1} (the initial value of A is $\{P_{n_i}, P_{n_{i+1}}, \ldots, P_{n_n}\}$). As we keep on adding the indices from A to P_{n+1} one by one, the datapoints corresponding to indices in P_{n+1} can be divided into three groups.

—Group 1 (G1) represents the datapoints, whose indices form the longest subsequence C of P_n, and whose elements are present in P_{n+1} (not necessarily in the same order). We denote this set of indices as $C = \text{LongestSubsequence}(P_n, P_{n+1})$. The LongestSubsequence procedure is described in Algorithm 4 and the five rows of Table II give five examples of this function. The initial value of C is set to $\{P_{n_1}, P_{n_2}, \ldots, P_{n_{i-1}}\}$. The remaining indices in P_n, which are not in C are represented by B ($B = P_n \backslash C$). B is initialized to $\{P_{n_i}, P_{n_{i+1}}, \ldots, P_{n_n}\}$. The MST connection indices of the datapoints in B is given by H, which is initialized to $\{F_{n_i}, F_{n_{i+1}}, \ldots, F_{n_n}\}$.
—Group 2 (G2) represents the new datapoint $\mathbf{x_{n+1}}$.
—Group 3 (G3) represents the datapoints corresponding to the remaining indices in P_{n+1}, which are not in G1 or G2. Let E represent the indices of the datapoints in this group. $E = P_{n+1} \backslash \{n+1\} \backslash C$ (the initial value of E is set to $\emptyset$).

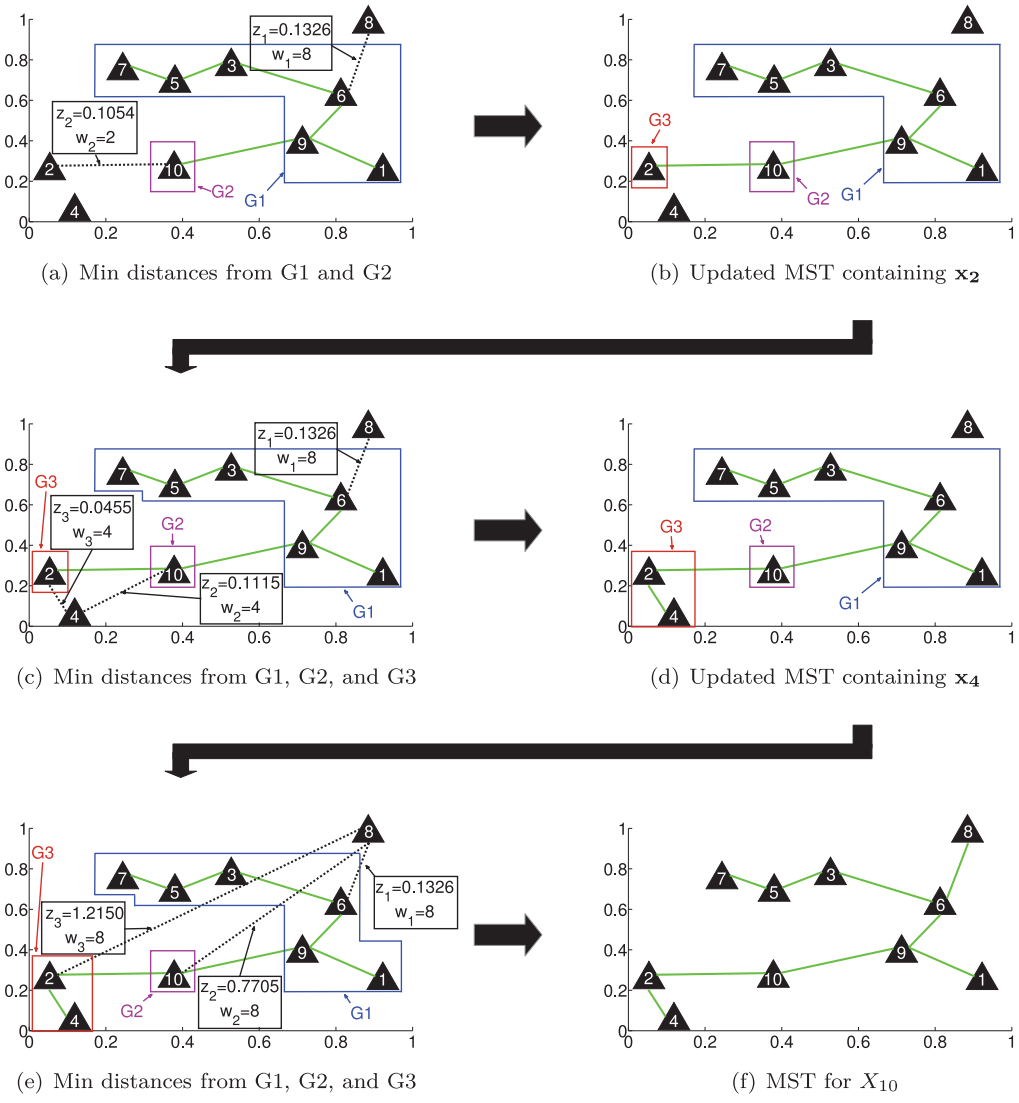
ALGORITHM 4: LongestSubsequence

Input : P_n − VAT reordering indices of D_n^*
P_{n+1} − VAT reordering indices of D_{n+1}^*
Output: LongestSubsequence(P_n, P_{n+1})

```
1 for j ← 1 to n do
2 |   if P_{n_j} ∉ P_{n+1} then
3 |   |   break
4 |   end
5 end
6 LongestSubsequence(P_n, P_{n+1}) = {P_{n_1}, P_{n_2}, ..., P_{n_{j-1}}}
```

The next index to be added to P_{n+1} corresponds to the datapoint represented by indices in A that is closest to any datapoint associated with the indices in P_{n+1}. Since we divided the datapoints corresponding to the indices in P_{n+1} into three groups as described above, we find the index of the closest datapoint from each of the three groups G1, G2, and G3. The closest datapoint from G1 can be found using the VAT property of P_n. Let w_1 be the index of this datapoint which is at a distance of z_1 from G1. The closest datapoint (having index w_2 and at a distance of z_2) from G2 can be found using Y and A. The closest datapoint from G3 can be found using a submatrix of the VAT reordered dissimilarity matrix D_n^* of X_n, whose rows and columns are given by indices in A and E, respectively. Let the closest datapoint from G3 have index w_3 and z_3 as its minimum distance. $v_i : 1 \leq i \leq 3$ represent the positions in P_{n+1} of the datapoints in G1, G2, and G3, which are closest to w_1, w_2, and w_3, respectively. The exact mathematical formulation to compute w_i, z_i, and v_i: $1 \leq i \leq 3$ is given in Section 5.3 and Algorithm 6 (lines 2–4). If $z_i = \min(z_j) : 1 \leq j \leq 3$, we add z_i to d_{n+1}, w_i to P_{n+1}, and v_i to F_{n+1}. This procedure is repeated until all the remaining datapoints (represented by indices in A) are added to P_{n+1}. Before giving the mathematical details of the inc-VAT algorithm, we demonstrate it for the 2D dataset of 10 points shown in Figure 6.

For the example shown in Figure 6(b), $A = \{8, 2, 4\}$, $G1 = \{\mathbf{x_1}, \mathbf{x_9}, \mathbf{x_6}, \mathbf{x_3}, \mathbf{x_5}, \mathbf{x_7}\}$, and their corresponding indices, $C = \{1, 9, 6, 3, 5, 7\}$, $B = \{8, 2, 4\}$, $G2 = \{\mathbf{x_{10}}\}$, and $G3 = E = \emptyset$. In Figures 7(a–e), G1 is in the blue box, G2 is in the magenta box, and G3 is in the red box, which are also appropriately marked as G1, G2, and G3, respectively. View (a) shows datapoints in G1 ($\{\mathbf{x_1}, \mathbf{x_9}, \mathbf{x_6}, \mathbf{x_3}, \mathbf{x_5}, \mathbf{x_7}\}$) enclosed in the blue box and those in G2 ($\{\mathbf{x_{10}}\}$) enclosed in the magenta box and since, $G3 = E = \emptyset$, there is no red box. From the VAT property of P_9, the index of the closest datapoint from G1 is $w_1 = 8$ at a distance of $z_1 = 0.1326$. Using Y and A, the closest datapoint from G2 has an index $w_2 = 2$ and distance $z_2 = 0.1054$. The datapoints in G1 and G2, which are closest to w_1 and w_2 are $\mathbf{x_6}$ and $\mathbf{x_{10}}$, respectively, and their position in P_{10} is given by $v_1 = 3$ and $v_2 = 7$. Two potential edges for MST of X_{10} are shown by two black dotted lines in Figure 7(a). Since $z_2 < z_1$, we add w_2 to the MST giving $P_{10} = \{1, 9, 6, 3, 5, 7, 10, 2\}$, $d_{10} = \{0.0622, 0.0659, 0.1033, 0.0283, 0.0223, 0.1272, 0.1054\}$, and $F_{10} = \{1, 1, 2, 3, 4, 5, 2, 7\}$ as shown in Figure 7(b). At this stage, C and B remain unchanged from the previous step, $A = \{8, 4\}$ and $G3 = \{\mathbf{x_2}\}$ with index $E = \{2\}$, which is shown by $\mathbf{x_2}$ being enclosed in a red box representing G3. The same procedure is repeated and indices of the new closest datapoints and their distances from G1, G2, and G3 are given by $w_1 = 8, z_1 = 0.1326$, $w_2 = 4, z_2 = 0.1115$, and $w_3 = 4, z_3 = 0.0455$, respectively, as shown by three dotted black edges in view (c). The datapoints in G1, G2, and G3, which are closest to w_1, w_2, and w_3 are $\mathbf{x_6}$, $\mathbf{x_{10}}$, and $\mathbf{x_2}$, respectively, and their positions in P_{10} are $v_1 = 3$, $v_2 = 7$, and $v_3 = 8$. We use the submatrix of the VAT reordered dissimilarity matrix D_9^* with rows corresponding to $G3 = \{\mathbf{x_2}\}$ and columns corresponding to $\{\mathbf{x_8}, \mathbf{x_4}\}$ (datapoints having



(a) Min distances from G1 and G2

(b) Updated MST containing $\mathbf{x_2}$

(c) Min distances from G1, G2, and G3

(d) Updated MST containing $\mathbf{x_4}$

(e) Min distances from G1, G2, and G3

(f) MST for X_{10}

Fig. 7. Example of the insertion scheme for incremental MST update.

indices given by $A = \{8, 4\}$) to find w_3 and z_3. Since the minimum of z_1, z_2, and z_3 is z_3, we add w_3 to the MST and get the updated MST as shown in Figure 7(d) with $P_{10} = \{1, 9, 6, 3, 5, 7, 10, 2, 4\}$, $d_{10} = \{0.0622, 0.0659, 0.1033, 0.0283, 0.0223, 0.1272, 0.1054, 0.0455\}$, and $F_{10} = \{1, 1, 2, 3, 4, 5, 2, 7, 8\}$. Now, $G3 = \{\mathbf{x_4}, \mathbf{x_2}\}$ as shown by the red box enclosing them in Figure 7(d). We apply the same procedure one more time as shown in view (e), where $w_1 = 8, z_1 = 0.1326, v_1 = 3, w_2 = 8, z_2 = 0.7705, v_2 = 7, w_3 = 8$, $z_3 = 1.2150$, and $v_3 = 8$. Three potential edges to be inserted in MST are shown by three dotted black lines in view (e). In this case $\min(z_1, z_2, z_3) = z_1$, hence we get the updated MST containing all the datapoints of X_{10} as shown in Figure 7(f) with inc-VAT outputs $P_{10} = \{1, 9, 6, 3, 5, 7, 10, 2, 4, 8\}$, $d_{10} = \{0.0622, 0.0659, 0.1033, 0.0283, 0.0223, 0.1272, 0.1054, 0.0455, 0.1326\}$, and $F_{10} = \{1, 1, 2, 3, 4, 5, 2, 7, 8, 3\}$.

5.3. inc-VAT Algorithm

Now, we give the detailed mathematical description and implementation of our inc-VAT algorithm (Algorithm 5) and efficient edge insertion scheme IncInsert (Algorithm 6). At any stage during updating the MST, let $C = \{P_{n_1}, P_{n_2}, \ldots, P_{n_s}\}, i \leq s \leq n$, where i is the index at which the insertion of $\mathbf{x_{n+1}}$ is made. In this case, $B = P_n \setminus C = \{P_{n_{s+1}}, P_{n_{s+2}}, \ldots, P_{n_n}\}$ and $H = \{F_{n_{s+1}}, F_{n_{s+2}}, \ldots, F_{n_n}\}$. The index of the closest datapoint from G1 is $w_1 = P_{n_{s+1}} = B_1$ and the minimum distance is $z_1 = d_{n_s} = d_{n_{Pos(B_1)-1}}$ (this follows from the VAT property of P_n), where $\text{Pos}(B_1)$ represents the position of index B_1 in P_n. The MST connection index of w_1 is $v_1 = \arg(P_{n+1} = P_{n_{H_1}})$. The closest point from the new datapoint $\mathbf{x_{n+1}}$ (G2) is $w_2 = A_{\text{argmin}(Y_A)}$ and the minimum distance is $z_2 = \min(Y_A)$. In this case, $v_2 = i$ because the MST connection index is the insertion position of the new datapoint, which is i. The closest point from G3 can be found using the submatrix of D_n^*, whose rows represent the datapoints corresponding to the third group of indices (represented by E) and columns represent the remaining datapoints to be added to P_{n+1} (their indices are represented by A). Let $z_3 = \min(D^*_{n_{\text{Pos}(A),\text{Pos}(E)}})$ be the minimum distance and (j, k) be the index of the closest points pair, where $\text{Pos}(A)$ and $\text{Pos}(E)$ represent the position of indices in A and E in P_n, respectively. The closest point and MST connection index from G3 are given by $w_3 = A_j$ and $v_3 = \arg(P_{n+1} = E_k)$. The vector G represents the reordering indices of P_n to obtain P_{n+1}.

ALGORITHM 5: inc-VAT

Input : $D_n^* - n \times n$ VAT reordered dissimilarity matrix for $X_n, n \geq 2$
$P_n -$ VAT reordering indices of D_n^*
$d_n -$ MST cut magnitude order of D_n^*
$F_n -$ MST connection indices of D_n^*
$V = \{v_1, v_2, \ldots, v_n\} -$ Distance of $\mathbf{x_{n+1}}$ from $\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$
Output: $D_{n+1}^* - (n+1) \times (n+1)$ VAT reordered dissimilarity matrix for X_{n+1}
$P_{n+1} -$ VAT reordering indices of D_{n+1}^*
$d_{n+1} -$ MST cut magnitude order of D_{n+1}^*
$F_{n+1} -$ MST connection indices of D_{n+1}^*

Find the insertion position i for $\mathbf{x_{n+1}}$
$(i, P_{n+1}, d_{n+1}, F_{n+1}) = \text{InsertPosition}(P_n, d_n, F_n, V)$ **(Algorithm 3)**
$A = \{P_{n_i}, P_{n_{i+1}}, \ldots, P_{n_n}\}$
$C = \{P_{n_1}, P_{n_2}, \ldots, P_{n_{i-1}}\}$
$B = P_n \setminus C$
$E = \emptyset$
$G = \{1, 2, \ldots, i-1\}$
$H = \{F_{n_i}, F_{n_{i+1}}, \ldots, F_{n_n}\}$
Reorder the remaining points after insertion position
while $A \neq \emptyset$ **do**
  $(A, B, E, P_{n+1}, d_{n+1}, F_{n+1}, G, H) = \text{IncInsert}(A, B, E, P_{n+1}, d_{n+1}, F_{n+1}, G, H, D_n^*, P_n, d_n, F_n)$ **(Algorithm 6)**
end
$D_n^* = D^*_{n_{G,G}}$
$Y^* = Y_G$
$Y^* = [Y_1^*, Y_2^*, \ldots, Y_{i-1}^*, 0, Y_i^*, Y_{i+1}^*, \ldots, Y_n^*]$
D_{n+1}^* = Insert Y^* after $i-1^{th}$ row and $i-1^{th}$ column of D_n^*

ALGORITHM 6: IncInsert

Input : $D_n^* - n \times n$ VAT reordered dissimilarity matrix for X_n
$P_n -$ VAT reordering indices of D_n^*
$d_n -$ MST cut magnitude order of D_n^*
$F_n -$ MST connection indices of D_n^*

Input-Output : $A, B, E, P_{n+1}, d_{n+1}, F_{n+1}, G, H$

$$\begin{cases} z_1 & = d_{n_{Pos(B_1)-1}} \\ w_1 & = B_1 \\ v_1 & = arg(P_{n+1} = P_{n_{H_1}}) \end{cases}$$ Minimum distance, closest point index, and MST connection index from G1

$$\begin{cases} z_2 & = min(Y_A) \\ w_2 & = A_{argmin(Y_A)} \\ v_2 & = i \end{cases}$$ Minimum distance, closest point index, and MST connection index from G2

$$\begin{cases} z_3 & = min(D^*_{n_{Pos(A),Pos(E)}}) \\ (j,k) & = argmin(D^*_{n_{Pos(A),Pos(E)}}) \\ w_3 & = A_j \\ v_3 & = arg(P_{n+1} = E_k) \end{cases}$$ Minimum distance, closest point index, and MST connection index from G3

$z = min(z_1, z_2, z_3)$

switch z **do**
  case z_1 **do**
    $(A, B, E, P_{n+1}, d_{n+1}, F_{n+1}, G, H) = \text{M1}(A, B, E, P_{n+1}, d_{n+1}, F_{n+1}, G, H, z_1, w_1, v_1)$ **(Algorithm 7)**
  end
  case z_2 **do**
    $(A, B, E, P_{n+1}, d_{n+1}, F_{n+1}, G, H) = \text{M2}(A, B, E, P_{n+1}, d_{n+1}, F_{n+1}, G, H, z_2, w_2, v_2)$ **(Algorithm 8)**
  end
  case z_3 **do**
    $(A, B, E, P_{n+1}, d_{n+1}, F_{n+1}, G, H) = \text{M3}(A, B, E, P_{n+1}, d_{n+1}, F_{n+1}, G, H, z_3, w_3, v_3)$ **(Algorithm 9)**
  end
end

The next index to be added to P_{n+1} is one of the three indices, w_1, w_2, and w_3. We find $z = \min(z_1, z_2, z_3)$ and based on which of the z_1, z_2, and z_3 is minimum, we apply three separate procedures M1, M2, and M3, respectively, to update $A, B, E, P_{n+1}, d_{n+1}, F_{n+1}, G$, and H.

If $z = z_1$, we use Procedure M1 as shown in Algorithm 7, where $\text{Pos}(w_1)$ denotes the position of w_1 in P_n.

If $z = z_2$, we use Procedure M2 as shown in Algorithm 8. If w_2 is the first element of A, LongestSubsequence(P_n, P_{n+1}) would change, hence, we update C, B, E, and H, otherwise we just append w_2 to E. Here also, $\text{Pos}(w_2)$ denotes the position of w_2 in P_n.

If $z = z_3$ (Procedure M3), we proceed similar to M2, except z_2, w_2, and v_2 are replaced by z_3, w_3, and v_3, respectively, as shown in Algorithm 9.

When all the remaining indices in P_n have been added to P_{n+1}, we have new reordering indices P_{n+1}, MST cut magnitude order d_{n+1}, and MST connection indices F_{n+1}. The next step is to get a new reordered dissimilarity matrix D^*_{n+1}. First, we reorder the rows and columns of D_n^* by the indices of G, per the new reordering indices P_{n+1}. We

ALGORITHM 7: Procedure M1

Input : z_1 − Minimum distance from G1
w_1 − Closest point index from G1
v_1 − MST connection index from G1
Input-Output : $A, B, E, P_{n+1}, d_{n+1}, F_{n+1}, G, H$
$P_{n+1} = \{P_{n+1}, w_1\}$; $d_{n+1} = \{d_{n+1}, z_1\}$; $F_{n+1} = \{F_{n+1}, v_1\}$
$G = \{G, Pos(w_1)\}$
$C =$ LongestSubsequence(P_n, P_{n+1}) **(Algorithm 4)**
$B = P_n \backslash C$
while $length(H) > length(B)$ **do**
  delete H_1
end
delete w_1 from A

ALGORITHM 8: Procedure M2

Input : z_2 − Minimum distance from G2
w_2 − Closest point index from G2
v_2 − MST connection index from G2
Input-Output : $A, B, E, P_{n+1}, d_{n+1}, F_{n+1}, G, H$
$P_{n+1} = \{P_{n+1}, w_2\}$; $d_{n+1} = \{d_{n+1}, z_2\}$; $F_{n+1} = \{F_{n+1}, v_2\}$
$G = \{G, Pos(w_2)\}$
if $w_2 = A_1$ **then**
  $C =$ LongestSubsequence(P_n, P_{n+1}) **(Algorithm 4)**
  $B = P_n \backslash C$
  while $length(H) > length(B)$ **do**
    delete H_1
  end
  $E = P_{n+1} \backslash \{n+1\} \backslash C$
else
  $E = \{E, w_2\}$
end
delete w_2 from A

also rearrange the distance of new point $\mathbf{x_{n+1}}$ to the previous points Y using the indices of G to obtain Y^*. The distance value of 0 is inserted at ith position of Y^* to account for the distance of $\mathbf{x_{n+1}}$ to itself. The vector Y^* is then inserted after $(i-1)$th row and $(i-1)$th column of D_n^* to obtain D_{n+1}^* as given in lines 14–17 of Algorithm 5.

5.4. inc-iVAT Algorithm

The inc-iVAT (Algorithm 10) produces the iVAT dissimilarity matrix D'^*_{n+1} for X_{n+1}. The submatrix consisting of the first $i-1$ rows and the first $i-1$ columns of D'^*_{n+1} is same as that of D'^*_n, where i is the insertion position of the new datapoint $\mathbf{x_{n+1}}$. The remaining elements of D'^*_{n+1} are computed using the same procedure as iVAT (Algorithm 2).

6. dec-VAT/dec-iVAT DESCRIPTION EXAMPLE AND ALGORITHM

For sliding window based clustering applications for time series data, it is not sufficient to just add the latest datapoint to the VAT-generated MST using inc-VAT/inc-iVAT as the memory required to store the reordered distance matrix will increase quadratically. An appropriate remedy is to keep removing the oldest datapoints from the MST

ALGORITHM 9: Procedure M3

Input : z_3 − Minimum distance from G3
w_3 − Closest point index from G3
v_3 − MST connection index from G3
Input-Output : $A, B, E, P_{n+1}, d_{n+1}, F_{n+1}, G, H$
$P_{n+1} = \{P_{n+1}, w_3\}$; $d_{n+1} = \{d_{n+1}, z_3\}$; $F_{n+1} = \{F_{n+1}, v_3\}$
$G = \{G, Pos(w_3)\}$
if $w_3 = A_1$ **then**
  $C = \text{LongestSubsequence}(P_n, P_{n+1})$ **(Algorithm 4)**
  $B = P_n \backslash C$
  while $length(H) > length(B)$ **do**
    delete H_1
  end
  $E = P_{n+1} \backslash \{n+1\} \backslash C$
else
  $E = \{E, w_3\}$
end
delete w_3 from A

ALGORITHM 10: inc-iVAT

Input : D^*_{n+1} − $(n+1) \times (n+1)$ inc-VAT reordered dissimilarity matrix for X_{n+1}
D'^*_n − $n \times n$ iVAT dissimilarity matrix for X_n
i − Insertion index of the new datapoint $\mathbf{x_{n+1}}$ in P_{n+1}
Output: D'^*_{n+1} − $(n+1) \times (n+1)$ inc-iVAT dissimilarity matrix for X_{n+1}
$c = \{1, 2, \ldots, i-1\}$
$D'^*_{n+1_{cc}} = D^*_{n_{cc}}$
for $r \leftarrow i$ **to** $n+1$ **do**
  $j = \arg \min_{1 \le m \le r-1} \{D^*_{n+1_{rm}}\}$
  $D'^*_{n+1_{rj}} = D^*_{n+1_{rj}}$
  $c = \{1, 2, \ldots, r-1\} \backslash \{j\}$
  $D'^*_{n+1_{rc}} = max\{D^*_{n+1_{rj}}, D'^*_{n+1_{jc}}\}$
  $D'^*_{n+1_{rc}} = D'^*_{n+1_{cr}}$
end

to keep array sizes manageable. This is achieved using the dec-VAT/dec-iVAT algorithm described in this section. dec-VAT implements a decremental version of VAT that removes a datapoint in the current MST using the properties of VAT (re)ordering as described in Section 5.

Consider a time series dataset $X_n = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$ having n datapoints arriving at time instants $\{1, 2, \ldots, n\}$. Let the dissimilarity matrix of dataset X_n be D_n. If we apply VAT to D_n, we get an $n \times n$ reordered dissimilarity matrix D^*_n, VAT reordering indices P_n, the MST cut magnitude order d_n, and the MST connection indices F_n. Let us assume that we want to remove a datapoint $\mathbf{x_p}$ from X_n, so that now we have modified dataset $X_{n-1} = X_n \backslash \{\mathbf{x_p}\}$. dec-VAT uses the VAT reordering information of X_n as a basis for a decremental version of VAT. Similar to inc-VAT, the dec-VAT algorithm is composed of two main steps:

(1) Finding the position of the to-be-removed datapoint $\mathbf{x_p}$ in the VAT ordering of X_n.
(2) Reordering the remaining datapoints after the deletion position.

We now describe each of these steps in more detail and formulate the dec-VAT and dec-iVAT algorithms.

6.1. Step 1: Finding the Position of the To-be-Removed Datapoint $\mathbf{x_p}$

The first step in dec-VAT is to determine the position i of $\mathbf{x_p}$ in P_n as given by lines 1 and 2 in Algorithm 11. Then, we determine the set of datapoints after position i in P_n that are connected to $\mathbf{x_p}$ in the current MST of X_n. This is achieved using F_n, the MST connection indices of P_n as given by lines 3 and 4 in Algorithm 11. Let these datapoints be represented by J. If $\mathbf{x_p}$ is a leaf node (the nodes that have only one branch edge), $J = \emptyset$, and in this case we just need to remove the node $\mathbf{x_p}$ from the MST of X_n using the procedure LeafNodeRemove (Algorithm 12) to obtain the VAT reordering of X_{n-1}. This is achieved by deleting the ith element (position of $\mathbf{x_p}$ in P_n) of P_n and F_n, and the $(i-1)$th element of d_n to obtain P_{n-1}, F_{n-1}, and d_{n-1}, respectively. Since we delete the i^{th} element $\mathbf{x_p}$, we decrease those values of F_{n-1} by 1 which are greater than i. We also delete the ith row and ith column of D_n^* to obtain D_{n-1}^*.

If $\mathbf{x_p}$ is not a leaf node (the nodes that have more than one branch edge), $J \neq \emptyset$, then we initialize P_{n-1}, d_{n-1}, and F_{n-1} based on whether the datapoint to be removed $\mathbf{x_p}$ is the first element of the VAT reordering of X_n (P_n) or not. If $\mathbf{x_p} = P_{n_1}$, we start the VAT reordering from the second element of P_n, P_{n_2} as given in lines 9–16 of Algorithm 11. If $\mathbf{x_p}$ is not the first element of P_n, we initialize P_{n-1} and F_{n-1} with the first $i-1$ elements of P_n and F_n, respectively. The MST cut magnitude order is initialized as $d_{n-1} = \{d_{n_1}, d_{n_2}, \ldots, d_{n_{i-2}}\}$. Similar to inc-VAT, $A = \{P_{n_{i+1}}, P_{n_{i+2}}, \ldots, P_{n_n}\}$ represents the remaining indices to be reordered and their MST connection indices are given by $H = \{F_{n_{i+1}}, F_{n_{i+2}}, \ldots, F_{n_n}\}$. $C = \text{LongestSubsequence}(P_n, \{P_{n-1}, i\})$ represents datapoints belonging to G1. The remaining indices in P_n, which are not in C, are given by $B = P_n \backslash C$. Since we are not adding any new datapoints here, there is no G2 ($G2 = \emptyset$). G3 represents the datapoints in P_{n-1}, which are not in G1. Let E represent the indices of the datapoints in this group, so that $E = \{P_{n-1}, i\} \backslash C$. The vector G represents the reordering of the indices of P_n to obtain P_{n-1}.

6.2. Step 2: Reordering the Remaining Datapoints After the Deletion Position

To insert the remaining elements having indices A, we use two procedures, SpecialInsert (Algorithm 13) and DecInsert (Algorithm 14) based on whether $B_1 = J_1$ or not. As we delete the datapoint $\mathbf{x_p}$ from the MST of D_n, we cut the edges joining $\mathbf{x_p}$ to the datapoints whose indices are given by J. The next nearest point to the current MST of X_{n-1} is provided by B_1. If B_1 is the same as the first datapoint in P_n, whose MST edge was cut, we can no longer use the VAT reordering because the connection point ($\mathbf{x_p}$) has been removed. In this case, we use the SpecialInsert procedure (Algorithm 13), which is just like the basic VAT algorithm. If this is not the case, we can still make use of the VAT reordering P_n, just like inc-VAT and call this procedure decInsert (Algorithm 14).

If $B_1 = J_1$, we find the distance of datapoints in A (indices of the remaining points) to the current points in P_{n-1} (whose rearranged order is given by G). In this case, the minimum distance (z), nearest datapoint (w), and MST connection index (v) are given by lines 2–5 of Algorithm 13 and they are appended to the appropriate matrices P_{n-1}, d_{n-1}, F_{n-1}, and G (lines 6–9). If $w \in J$, then we delete w from A and J, otherwise add it to E. Additionally if $w = J_1$, we update C, B, and H (lines 13–17) as the datapoint attached to the deleted node of the MST of X_n is reconnected to the MST of X_{n-1}.

If $B_1 \neq J_1$, we can insert the remaining datapoints whose indices are given by A into the MST of X_{n-1} using the same procedure that was used in inc-VAT (using G1, G2, and G3). In this case, since there is no new datapoint, we do not have G2. This procedure (DecInsert) is described in Algorithm 14 and uses procedures M1 (Algorithm 7) and M3 (Algorithm 9) to update P_{n-1}, d_{n-1}, and F_{n-1}.

ALGORITHM 11: dec-VAT

Input : $D_n^* - n \times n$ VAT reordered dissimilarity matrix for X_n
$P_n -$ VAT reordering indices of D_n^*
$d_n -$ MST cut magnitude order of D_n^*
$F_n -$ MST connection indices of D_n^*
$\mathbf{x_p} -$ Point to remove
Output: $D_{n-1}^* - (n-1) \times (n-1)$ VAT reordered dissimilarity matrix for X_{n-1}
$P_{n-1} -$ VAT reordering indices of D_{n-1}^*
$d_{n-1} -$ MST cut magnitude order of D_{n-1}^*
$F_{n-1} -$ MST connection indices of D_{n-1}^*

Find the position i **of** $\mathbf{x_p}$ **in** P_n
$i = arg(P_n = p)$
Find the datapoints (J) **and their indices** (I) **in** P_n**, that are connected to** $\mathbf{x_p}$ **in the MST of** D_n^*
$I = arg(F_n = i)$; $J = P_{n_I}$
if $J = \emptyset$ **then**
  $(D_{n-1}^*, P_{n-1}, d_{n-1}, F_{n-1}) =$ LeafNodeRemove$(D_n^*, P_n, d_n, F_n, i)$ **(Algorithm 12)**
else
  if $i = P_{n_1}$ **then**
    $P_{n-1} = \{P_{n_2}\}$; $d_{n-1} = \emptyset$; $F_{n-1} = \{1\}$
    $A = \{P_{n_3}, P_{n_4}, \ldots, P_{n_n}\}$
    $C = \{P_{n_1}, P_{n_2}\}$
    $B = P_n \backslash C$
    $E = \emptyset$
    $G = \{2\}$
    $H = \{F_{n_3}, F_{n_4}, \ldots, F_{n_n}\}$
    $k = arg(J = P_{n_2})$; delete J_k
  else
    $P_{n-1} = \{P_{n_1}, P_{n_2}, \ldots, P_{n_{i-1}}\}$; $d_{n-1} = \{d_{n_1}, d_{n_2}, \ldots, d_{n_{i-2}}\}$; $F_{n-1} = \{F_{n_1}, F_{n_2}, \ldots, F_{n_{i-1}}\}$
    $A = \{P_{n_{i+1}}, P_{n_{i+2}}, \ldots, P_{n_n}\}$
    $C = \{P_{n_1}, P_{n_2}, \ldots, P_{n_i}\}$
    $B = P_n \backslash C$
    $E = \emptyset$
    $G = \{1, 2, \ldots, i-1\}$
    $H = \{F_{n_{i+1}}, F_{n_{i+2}}, \ldots, F_{n_n}\}$
  end
  while $A \neq \emptyset$ **do**
    if $B_1 = J_1$ **then**
      $(A, B, E, P_{n-1}, d_{n-1}, F_{n-1}, G, H, J) =$
      SpecialInsert$(A, B, E, P_{n-1}, d_{n-1}, F_{n-1}, G, H, J, D_n^*, P_n, d_n, F_n)$ **(Algorithm 13)**
    else
      $(A, B, E, P_{n-1}, d_{n-1}, F_{n-1}, G, H) =$
      DecInsert$(A, B, E, P_{n-1}, d_{n-1}, F_{n-1}, G, H, D_n^*, P_n, d_n, F_n)$ **(Algorithm 14)**
    end
  end
  $D_{n-1}^* = D_{n_{G,G}}^*$
end

6.3. dec-iVAT Algorithm

The dec-iVAT (Algorithm 15) provides the iVAT dissimilarity matrix $D_{n-1}'^*$ of X_{n-1}. The procedure is similar to inc-iVAT (Algorithm 10), where the submatrix consisting of the first $i-1$ rows and the first $i-1$ columns of $D_{n-1}'^*$ is same as that of $D_n'^*$, where i is

ALGORITHM 12: LeafNodeRemove

Input : $D_n^* - n \times n$ VAT reordered dissimilarity matrix for X_n
$P_n -$ VAT reordering indices of D_n^*
$d_n -$ MST cut magnitude order of D_n^*
$F_n -$ MST connection indices of D_n^*
$i -$ Position of leaf node $\mathbf{x_p}$ in P_n
Output: $D_{n-1}^* - (n-1) \times (n-1)$ VAT reordered dissimilarity matrix for X_{n-1}
$P_{n-1} -$ VAT reordering indices of D_{n-1}^*
$d_{n-1} -$ MST cut magnitude order of D_{n-1}^*
$F_{n-1} -$ MST connection indices of D_{n-1}^*

Initialize P_{n-1}, F_{n-1}, d_{n-1}, **and** D_{n-1}^*
$P_{n-1} = P_n$
$F_{n-1} = F_n$
$d_{n-1} = d_n$
$D_{n-1}^* = D_n^*$
Delete the elements corresponding to $\mathbf{x_p}$
delete P_{n-1_i}
delete F_{n-1_i}
$K = arg(F_{n-1} > i)$
$F_{n-1_K} = F_{n-1_K} - 1$
delete $d_{n-1_{i-1}}$
delete i^{th} row and i^{th} column of D_{n-1}^*

ALGORITHM 13: SpecialInsert

Input : $D_n^* - n \times n$ VAT reordered dissimilarity matrix for X_n
$P_n -$ VAT reordering indices of D_n^*
$d_n -$ MST cut magnitude order of D_n^*
$F_n -$ MST connection indices of D_n^*
Input-Output : $A, B, E, P_{n-1}, d_{n-1}, F_{n-1}, G, H, J$
$z = min(D^*_{n_{G,Pos(A)}})$
$(j, k) = argmin(D^*_{n_{G,Pos(A)}})$
$w = A_j$
$v = arg(P_{n-1} = G_k)$
$P_{n-1} = \{P_{n-1}, w\}$
$d_{n-1} = \{d_{n-1}, z\}$
$F_{n-1} = \{F_{n-1}, v\}$
$G = \{G, Pos(w)\}$
if $w \in J$ **then**
  delete w from A
  if $w = J_1$ **then**
    $C =$ LongestSubsequence$(P_n, \{P_{n-1}, i\})$ **(Algorithm 4)**
    $B = P_n \backslash C$
    while $length(H) > length(B)$ **do**
      delete H_1
    end
  end
  delete w from J
else
  $E = \{E, w\}$
end

ALGORITHM 14: DecInsert

Input : $D_n^* - n \times n$ VAT reordered dissimilarity matrix for X_n
$P_n -$ VAT reordering indices of D_n^*
$d_n -$ MST cut magnitude order of D_n^*
$F_n -$ MST connection indices of D_n^*

Input-Output : $A, B, E, P_{n-1}, d_{n-1}, F_{n-1}, G, H$

$$\begin{cases} z_1 & = d_{n_{Pos(B_1)-1}} \\ w_1 & = B_1 \\ v_1 & = arg(P_{n-1} = P_{n_{H_1}}) \end{cases}$$ Minimum distance, closest point index, and MST connection index from G1

$$\begin{cases} z_3 & = min(D^*_{n_{Pos(A),Pos(E)}}) \\ (j,k) & = argmin(D^*_{n_{Pos(A),Pos(E)}}) \\ w_3 & = A_j \\ v_3 & = arg(P_{n-1} = E_k) \end{cases}$$ Minimum distance, closest point index, and MST connection index from G3

$z = min(z_1, z_3)$
switch z **do**
  case z_1 **do**
    $(A, B, E, P_{n-1}, d_{n-1}, F_{n-1}, G, H) = \text{M1}(A, B, E, P_{n-1}, d_{n-1}, F_{n-1}, G, H, z_1, w_1, v_1)$ **(Algorithm 7)**
  end
  case z_3**do**
    $(A, B, E, P_{n-1}, d_{n-1}, F_{n-1}, G, H) = \text{M3}(A, B, E, P_{n-1}, d_{n-1}, F_{n-1}, G, H, z_3, w_3, v_3)$ **(Algorithm 9)**
  end
end

the position of the deleted datapoint $\mathbf{x_p}$ in P_n. The remaining elements of D'^*_{n-1} are computed using the same procedure as iVAT (Algorithm 2).

7. TIME COMPLEXITY

The tendency assessment problem consists of two steps: the first step calculates the $n \times n$ dissimilarity matrix of n datapoints at a particular instant of time. This has a time complexity of $O(n^2)$ as $\frac{n \times (n-1)}{2}$ pairwise distances need to be computed. The next step is to reorder the dissimilarity matrix based on the EMST indices of the datapoints. For streaming datasets, we need to update the dissimilarity matrix to account for insertion and deletion of the datapoints, which can be performed in $O(n)$ time as we just need to calculate the distance of the new point to the previous points. We also need to update the reordering to reflect changes in the updated EMST.

The new algorithms, inc-VAT and dec-VAT, use the VAT reordering property of X_n to the full extent to update the EMST for X_{n+1} and X_{n-1}, respectively. Hence, for most cases, their time complexities are much less than that of VAT, which has a time complexity of $O(n^2)$. This update operation usually influences a limited number of the closest neighbors only. However, depending on the current structure of the MST, an insertion or deletion can (although rarely) change the complete structure of the MST, hence has a worst case time complexity of $O(n^2)$ as discussed in Sections 7.1 and 7.2.

The issue of traditional approaches to algorithm analysis using the running time for the worst possible input as an upper bound for the running time of all instances was also raised in March et al. [2010]. This often leads to overly pessimistic bounds due to a few pathological inputs. The adaptive analysis provides a solution to improve

ALGORITHM 15: dec-iVAT

Input : D^*_{n-1} – $(n-1)\times(n-1)$ dec-VAT reordered dissimilarity matrix for X_{n-1}
D'^*_n – $n \times n$ iVAT dissimilarity matrix for X_n
i – Index of the deleted datapoint $\mathbf{x_p}$ in P_n
Output: D'^*_{n-1} – $(n-1)\times(n-1)$ dec-iVAT dissimilarity matrix for X_{n-1}

$c = \{1, 2, \ldots, i-1\}$
$D'^*_{n-1_{cc}} = D'^*_{n_{cc}}$
for $r \leftarrow i$ **to** $n-1$ **do**
  $j = \arg\min_{1\le m\le r-1}\{D^*_{n-1_{rm}}\}$
  $D'^*_{n-1_{rj}} = D^*_{n-1_{rj}}$
  $c = \{1, 2, \ldots, r-1\}\backslash\{j\}$
  $D'^*_{n-1_{rc}} = max\{D^*_{n-1_{rj}}, D'^*_{n-1_{jc}}\}$
  $D'^*_{n-1_{rc}} = D'^*_{n-1_{cr}}$
end

these results by considering properties of the inputs in the time complexity analysis. By bounding the runtime in terms of these features, one can obtain tighter and more informative bounds. Adaptive analysis has been successfully applied to many fundamental problems including searching in lists [Bentley and Yao 1976], merging arrays [Demaine et al. 2000], sorting [Estivill-Castro and Wood 1992], and the convex hull problem [Kirkpatrick and Seidel 1986]. Despite these successes, the difficulty of characterizing the inputs with the problem has limited the number of applications. For the algorithms introduced in this article, the time complexity for inserting a new datapoint into the MST or deleting an existing point depends on the complete structure of the points in the dataset and the new datapoint. Hence, it is not possible to characterize the time complexity in terms of the input dataset structure.

Although inc-VAT/dec-VAT and inc-iVAT/dec-iVAT significantly lower the time complexity of VAT and iVAT, respectively, the output of all of these algorithms is a reordered distance matrix that has n^2 elements, so storing and visualizing them could be problematic due to software and hardware constraints as n becomes large. The practical limit on dataset size, n, is system and platform dependent. In this article, we set the maximum practical limit of n to be 5,000. All the programs are written in MATLAB 2012. The computational platform is OS: Windows 7 (64bit); Processor: Intel(R) Core(TM) i7-2600 @3.40GHz; RAM: 8GB.

7.1. inc-VAT/inc-iVAT vs. VAT/iVAT

The first step of inc-VAT, which is to find the insertion position i of the new datapoint $\mathbf{x_{n+1}}$, requires a maximum of n comparisons and hence has a time complexity of $O(n)$. Next, we need to rearrange the remaining $n-(i-1)$ datapoints of X_n, each of which uses one of the three procedures M1, M2, or M3 based on which of z_1, z_2, or z_3 is minimum. Finding z_1, w_1, and v_1 and executing M1 can be done in constant time as it just involves finding the first element of B. Finding z_2, w_2, and v_2 and executing M2 requires a maximum of $|A|$ comparisons, where $|A|$ is the length of A, which decreases from $n-(i-1)$ to 0 as we keep on adding datapoint indices to P_{n+1}. Finding z_3, w_3, and v_3 and executing M3 requires a maximum of $|A|\times|E|$ comparisons, where $|A|$ and $|E|$ represent the length of A and E, respectively, and are (usually) much smaller than n.

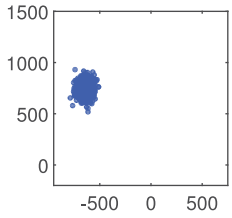
inc-iVAT (re)uses part of the iVAT dissimilarity matrix from the previous step, D'^*_n, and hence takes less time than iVAT. The time taken by inc-iVAT depends on the insertion index i of the new datapoint $\mathbf{x_{n+1}}$ in P_{n+1}. The higher the value of i, the larger

the portions of the iVAT matrix from the previous step that can be reused and hence less time is taken by inc-iVAT. If i is small, inc-iVAT takes more time, and in this case, the time taken by inc-VAT is also large as $(n - (i - 1))$ indices need to be rearranged. It is not possible to find the theoretical bound on the time complexities of the inc-VAT and inc-iVAT algorithms as it depends on the current MST of the dataset and distance of the new datapoint from all the datapoints in the dataset. Although the worst case time complexity of inc-iVAT is $O(n^2)$, for practical applications ($n \leq 5{,}000$), the time taken by inc-iVAT is much less than that taken by inc-VAT. Hence, the time complexity of $O(n^2)$ is acceptable, as shown in the illustration below.

To illustrate the effectiveness of inc-VAT/inc-iVAT to demonstrate the evolving cluster structure in a dataset, we performed an experiment on a 2D Gaussian mixture of five clusters called X that has a total of 5,000 datapoints. The number of datapoints in each of the five clusters is 1,300, 1,000, 400, 1,600, and 700, respectively. The datapoints in X are arranged according to the cluster they belong. Hence, the first 1,300 rows of X are x and y coordinates of the datapoints belonging to the first cluster, the next 1,000 rows represent the datapoints belonging to the second cluster and so on.

We start with the first two datapoints and add one datapoint at a time. At each step, the inc-VAT and inc-iVAT algorithms are executed. Figures 8(a–c) show a subset of X consisting of the first 1,300 datapoints (all belonging to first cluster), called $X_{1,300}$ and its (incrementally built) inc-VAT and inc-iVAT images. The presence of a single dark block in views (b) and (c) of Figure 8 confirms the presence of a single cluster in $X_{1,300}$. We keep on adding one datapoint at a time to the MST using inc-VAT and inc-iVAT algorithms until we have 2,300 datapoints belonging to first and second cluster ($X_{2,300}$). Figures 8(d–f) show 2D scatterplot and incrementally built inc-VAT and inc-iVAT images for $X_{2,300}$. The inc-VAT and inc-iVAT images of view (e) and (f) suggest the presence of two clusters in $X_{2,300}$. This process is continued until all the datapoints in X are included in the MST. Figures 8(g–i), (j–l), and (m–o) show the scatterplot, inc-VAT, and inc-iVAT image for $X_{2,700}$ (containing first three clusters), $X_{4,300}$ (containing first four clusters), and $X_{5,000}$ (containing all the five clusters), respectively. The inc-VAT and inc-iVAT images for $X_{2,700}$, $X_{4,300}$, and $X_{5,000}$ show the presence of three, four, and five clusters, respectively, by three, four, and five dark blocks along the diagonal.

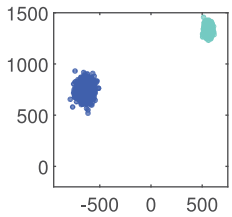
To illustrate the time complexities of VAT/iVAT and inc-VAT/inc-iVAT, we perform an experiment on the same 2D Gaussian mixture X, but we randomize the rows of X so that datapoints belonging to the same cluster are not adjacent anymore. We start with the first two datapoints and add one datapoint at a time. At each step, the VAT, iVAT, inc-VAT, and inc-iVAT algorithms are executed and the time required to compute the respective reordered dissimilarity matrices is recorded. We also compare our run time results with the optimal dual-tree Boruvka algorithm [March et al. 2010] for EMST computation. Figure 9(a) shows the time taken by VAT (black curve), inc-VAT (green curve), and the dual-tree Boruvka algorithm (red curve). View (a) confirms that the VAT time complexity (the black curve, which is almost a perfect parabola through the origin) increases quadratically as $O(n^2)$ and the dual-tree Boruvka algorithm time increases as $O(n \times \log(n))$. Whereas, the time taken by inc-VAT is much smaller than VAT and dual-tree Boruvka algorithm. Figure 9(b) shows the time comparison between iVAT (black curve) and inc-iVAT (green curve). Since inc-iVAT reuses part of the iVAT/inc-iVAT matrix from the previous step, its time complexity is always less than that of iVAT, which is also evident from view (b) as the black curve (iVAT) forms as an upper envelope for the green curve (inc-iVAT). Figure 9(c) shows the time comparison of inc-VAT and inc-iVAT, which suggests that the time taken by inc-iVAT is less than that taken by inc-VAT, and hence, is not a limiting factor for practical purposes ($n \leq 5{,}000$). Finally, Figure 9(d) shows the time taken to generate the iVAT image from the input distance matrix using VAT and iVAT (black curve) and the time required for EMST



(a) $X_{1,300}$

(b) inc-VAT image of $X_{1,300}$

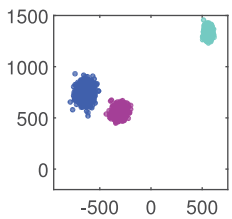
(c) inc-iVAT image of $X_{1,300}$



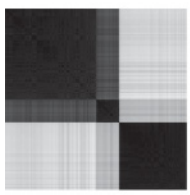
(d) $X_{2,300}$

(e) inc-VAT image of $X_{2,300}$

(f) inc-iVAT image of $X_{2,300}$

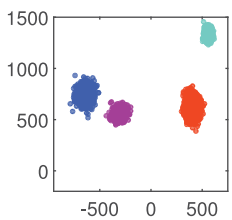


(g) $X_{2,700}$

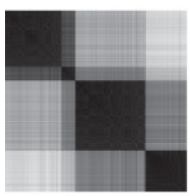


(h) inc-VAT image of $X_{2,700}$

(i) inc-iVAT image of $X_{2,700}$

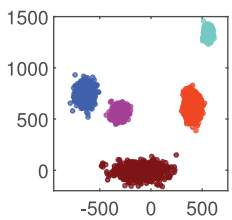


(j) $X_{4,300}$

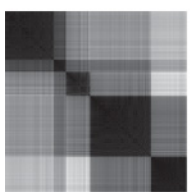


(k) inc-VAT image of $X_{4,300}$

(l) inc-iVAT image of $X_{4,300}$



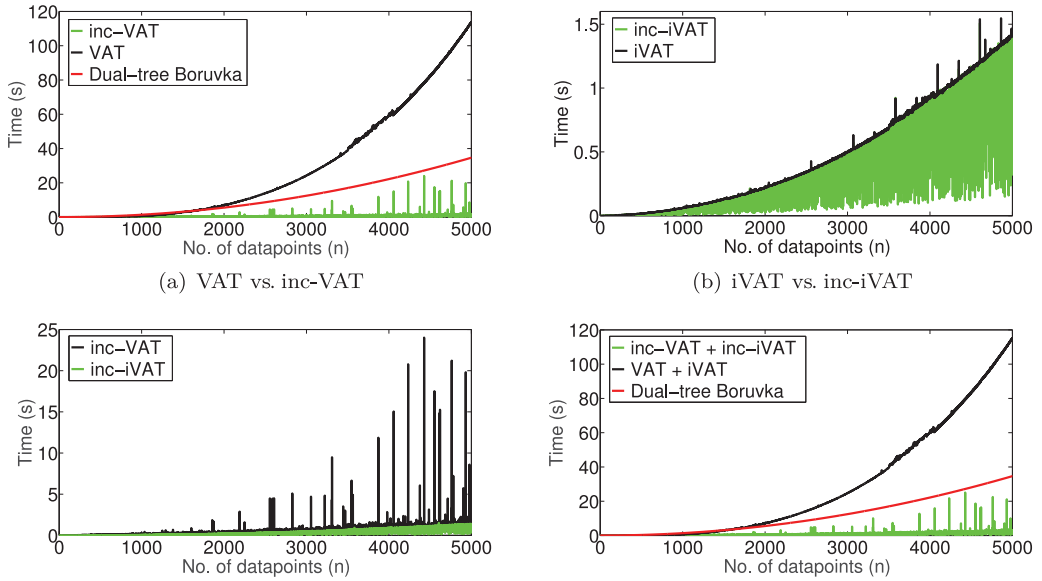
(m) $X_{5,000}$



(n) inc-VAT image of $X_{5,000}$

(o) inc-iVAT image of $X_{5,000}$

Fig. 8. 2D data scatterplot and (incrementally built) inc-VAT and inc-iVAT images of X at $n = 1, 300, 2{,}300, 2{,}700, 4{,}300$, and $5{,}000$.



(a) VAT vs. inc-VAT (b) iVAT vs. inc-iVAT

(c) inc-VAT vs. inc-iVAT (d) VAT + iVAT vs. inc-VAT + inc-iVAT

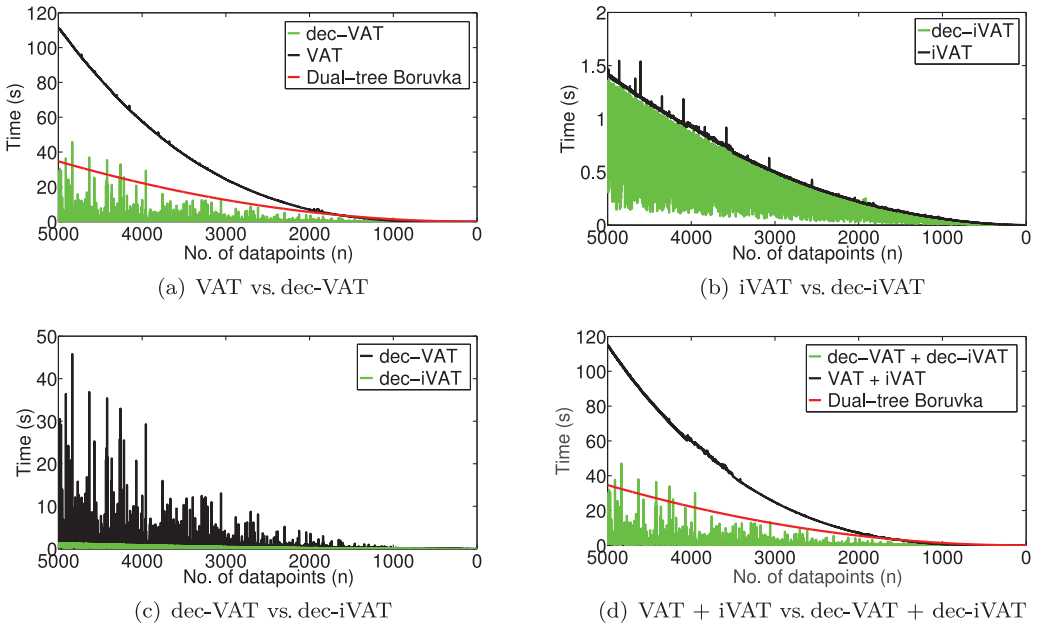
Fig. 9. Time comparison of VAT, iVAT, inc-VAT, and inc-iVAT algorithms for the 5,000 point 2D dataset.

computation using the dual-tree Boruvka algorithm and subsequent distance matrix reordering (red curve). The time taken to update the inc-iVAT image is shown by green curve, representing inc-VAT + inc-iVAT. View (d) clearly shows that the time required to update the inc-iVAT image is much less than that needed to generate a new image using VAT/iVAT or dual-tree Boruvka algorithm as n increases.

7.2. dec-VAT/dec-iVAT vs. VAT/iVAT

Similar to inc-VAT, the first step of dec-VAT is to find the position i of the to-be-removed datapoint $\mathbf{x_p}$ in P_n, which requires a maximum of n comparisons and hence has a time complexity of $O(n)$. Next, we find the datapoints connected to $\mathbf{x_p}$ in the MST of X_n, which again requires a maximum of n comparisons having a time complexity of $O(n)$. If $\mathbf{x_p}$ is a leaf node of the MST, we just need to remove it from the MST of X_n to obtain the VAT reordering of X_{n-1}, which can be done in constant time. If $\mathbf{x_p}$ is not a leaf node, we need to insert the remaining datapoints using one of the procedures, SpecialInsert or DecInsert. SpecialInsert is a VAT type of insertion having a time complexity of $O(|G| \times |A|)$, which needed to be executed a minimum of $|J|$ times, which is much smaller than n. DecInsert is an IncInsert type of insertion requiring one of the two procedures, M1 or M3, based on which of z_1 or z_3 is minimum. The time complexity of DecInsert is the same as that of IncInsert as described in Section 7.1.

To illustrate the time complexity difference between dec-VAT/dec-iVAT, VAT/iVAT, and the dual-tree Boruvka algorithm, we perform an experiment on the 2D Gaussian mixture X shown in Figure 8(m). We start with $n = 5{,}000$ datapoints and remove one randomly chosen datapoint at a time. At each step, the VAT, iVAT, dec-VAT, dec-iVAT, and the dual-tree Boruvka algorithms are executed and the time required to compute the respective reordered dissimilarity matrices is recorded. Figure 10(a) shows the time taken by VAT (black curve), dec-VAT (green curve), and the dual-tree Boruvka MST computation followed by reordering the distance matrix (red curve). Figure 10(b) shows the time comparison between iVAT (black curve) and dec-iVAT (green curve).



(a) VAT vs. dec-VAT (b) iVAT vs. dec-iVAT

(c) dec-VAT vs. dec-iVAT (d) VAT + iVAT vs. dec-VAT + dec-iVAT

Fig. 10. Time comparison of VAT, iVAT, dec-VAT, and dec-iVAT algorithms for the 5,000 point 2D dataset.

Since dec-iVAT reuses part of the iVAT/dec-iVAT matrix from the previous step, its time complexity is always less than that of iVAT, which is also evident from view (b) as the black curve (iVAT) forms as an upper envelope for the green curve (dec-iVAT). Figure 10(c) shows the time comparison of dec-VAT and dec-iVAT, which suggests that the time taken by dec-iVAT is less than that taken by dec-VAT, and hence, is not a limiting factor for practical purposes ($n \leq 5{,}000$). Finally, Figure 10(d) shows the time taken to generate the iVAT image from the distance matrix using VAT and iVAT (black curve) and the dual-tree Boruvka algorithm (red curve). The time required to update the dec-iVAT image is shown by the green curve representing dec-VAT + dec-iVAT. View (d) clearly shows that the time needed to update the dec-iVAT image is (for most of the cases) much less than that needed to generate a new iVAT image using VAT or dual-tree Boruvka algorithm as n increases.

7.3. Sliding Window Based Time Series Clustering

In this experiment, we apply inc-iVAT/dec-iVAT to a time series data with a variable sliding time window, having window sizes of $n = 100, 200, \ldots,$ up to 5,000. For each window size, n, VAT/iVAT and the dual-tree Boruvka algorithms are applied to an $n \times n$ distance matrix of n 2D datapoints. For each window size, n, we insert a new datapoint using the inc-VAT/inc-iVAT algorithm and delete one datapoint using the dec-VAT/dec-iVAT algorithm. This process is repeated 500 times for each window size and the average time to perform one each of inc-VAT, inc-iVAT, dec-VAT, and dec-iVAT operations is calculated. Figure 11 plots the time needed to generate the (static) RDI using VAT/iVAT (black curve) and the dual-tree Boruvka algorithm (red curve). The average time over 500 operations to incrementally update the VAT/iVAT image using inc-VAT, inc-iVAT, dec-VAT, and dec-iVAT is given by the green curve. It is apparent from Figure 11 that the time taken to update an iVAT image using the inc-VAT, inc-iVAT, dec-VAT, and dec-iVAT algorithms is much less than the time needed to generate a new one using VAT/iVAT or dual-tree Boruvka algorithms as n increases.

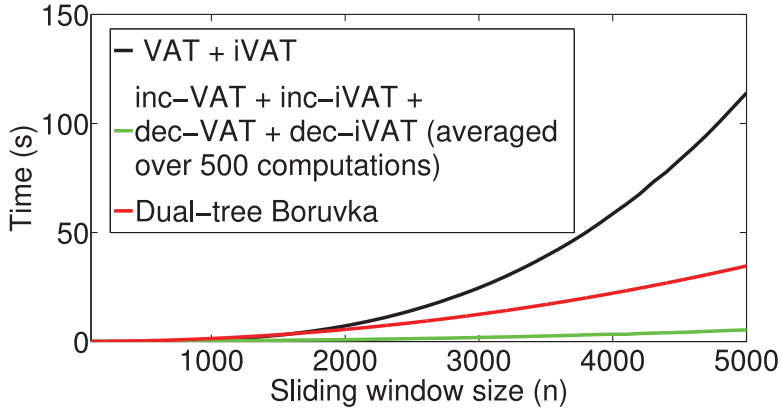


Fig. 11. Time comparison between generating vs. updating an iVAT image for a sliding time window.

8. ANOMALY DETECTION AND ONLINE CLUSTERING USING inc-iVAT/dec-iVAT ALGORITHMS

Anomaly detection can be regarded as a particular case of data clustering, wherein clusters that are too far from the main clusters, or have too few datapoints, are the anomalies. We use this concept for online anomaly detection and clustering in streaming data. Clustering-based anomaly detection has been studied previously in He et al. [2003], where the clusters are first labeled as large and small depending on the number of datapoints in them. A *cluster-based local outlier factor* (CBLOF) is then defined for all the datapoints based on the size of each cluster and its nearest cluster. Whenever a new datapoint is added, or an old datapoint is removed from the MST using inc-iVAT/dec-iVAT, we use a slight variation of the iVAT+ anomaly detection framework [Kumar et al. 2015] for automatically detecting anomalies from the updated iVAT image. The iVAT+ framework consists of the base model *plus* the heuristics in Equations (3) and (4) (described below), which are added to the basic model to make it applicable to anomaly detection and clustering. Clusters in the "+" model are obtained by cutting the MST using threshold magnitudes ordered by edge distances d_n in the MST. To make the cluster boundary threshold independent of the data, the threshold in iVAT+ [Kumar et al. 2015] is modified as follows: The cluster boundaries are defined by those indices t, which satisfy

$$d_{n_t} > \alpha \times \text{mean}(d_n), \tag{3}$$

where $\alpha \geq 1$ is a user-defined parameter that controls how far two groups of datapoints should be from each other to be considered as separate clusters. Hence, we cut those edges of the MST generated by inc-iVAT/dec-iVAT that satisfy Equation (3), resulting in one or more connected subtrees (clusters), which are denoted by $X_i, i \geq 1$. Small values of α create tighter boundaries for clusters, and large values of α represent loose cluster boundaries.

We use the number of datapoints in a cluster as a measure of normality for the cluster. A cluster X_i is regarded as an anomalous set of points if it satisfies the following condition:

$$|X_i| < \lfloor \beta \times n \rfloor, \tag{4}$$

where $|X_i|$ is the number of datapoints in cluster X_i, $0 \leq \beta \leq 1$ is a user-defined parameter, and n is the total number of points in the dataset. Large values of β (close to 1) cause even large groups of datapoints that are far from other clusters to be declared anomalous. As β decreases, the same abnormal cluster eventually becomes part of the

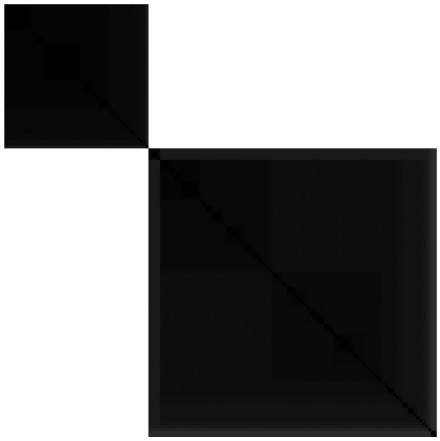


Fig. 12. iVAT reordered dissimilarity image for $IRIS_{150}$.

regular cluster structure, and only isolated datapoints or a group of a few datapoints remain anomalous.

9. NUMERICAL EXPERIMENTS

In this section, we describe various numerical experiments to show the performance of inc-iVAT and dec-iVAT for anomaly detection and sliding window based data clustering. We have performed anomaly detection experiments on three datasets, IRIS [Anderson 1935], Heron Island weather station [Bezdek et al. 2011], and the REDUCE energy dataset [Murtagh et al. 2013]. The sliding window based data clustering experiment is performed on the IoT data from an environment sensing unit employed at two locations in the city of Melbourne [Internet of Things: A pilot deployment in the city of Melbourne, Australia 2015].

9.1. IRIS Data Experiment

In this experiment, we deliberately inserted five anomalies into the IRIS data to see whether inc-VAT/inc-iVAT spots them as they occur. We begin with the IRIS data in its standard form (I.Setosa followed by I.Versicolor followed by I.Virginica), which gives 150 4D datapoints (denoted by $IRIS_{150}$). Figure 12 shows the iVAT RDI for $IRIS_{150}$. The principal diagonal consists of two dark blocks, of which the smaller one corresponds to I.Setosa (50 datapoints) and the bigger block corresponds to I.Versicolor and I.Virginica (100 datapoints).

Next, we inserted five abnormalities at random locations in the IRIS dataset. Specifically, we inserted (10,10,10,10), (−10,10,10,10), (−9.6,10.2,−10.7,8.9), (10,10.2,9.8,10), and (−10,10.2,9.8,10) at locations 21, 35, 70, 94, and 130, respectively. For example, location 21 means that the first random point is inserted into IRIS after datapoint 20 in the original order, and so on. We call this augmented dataset $IRIS_{155}$. For this demonstration, we have set the parameters $\alpha = 10$ and $\beta = 0.02$. The reason for the choice of α and β is discussed later in the section.

When the $IRIS_{155}$ is processed by inc-VAT/ inc-iVAT point by point, we can identify all the anomalies. Figure 13 shows the d_n plot and inc-iVAT image before and after the anomaly at position 21. After the 21st datapoint is processed, the d_n value shoots above the threshold value (red horizontal line) and hence is an anomaly as shown by the isolated red pixel (encircled for clarity) at the bottom right corner of the inc-iVAT image.

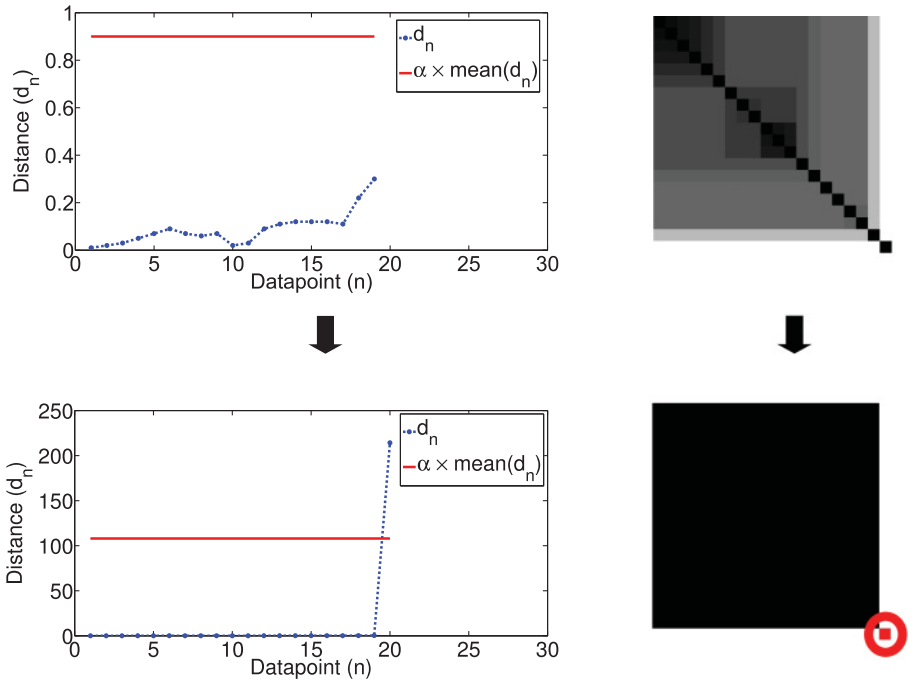


Fig. 13. d_n plot and inc-iVAT image for datapoints 20 (top) and 21 (bottom) on $IRIS_{155}$.

Similarly, Figure 14 shows the d_n plot and the inc-iVAT image before and after the anomaly at position 35. Exactly as above, as the 35th datapoint is processed, the d_n value shoots above the threshold value and hence the second anomaly is shown by the green pixel (encircled) at the bottom-right corner of the inc-iVAT image. This inc-iVAT image includes the previously detected anomalous datapoint at location 21, which is the red pixel adjacent to and above the green pixel.

Figure 15 shows the zoomed right-bottom corner of the inc-iVAT images just before and after the anomalies at the 70th, 94th, and 130th positions. Anomaly 3 at the 70th position is quite displaced from the first two anomalies, so results in a third isolated magenta pixel appearing in Figure 15(d). On the other hand, anomaly 1 (10,10,10,10) and anomaly 4 (10,10.2,9.8,10), and anomaly 2 (−10,10,10,10) and anomaly 5 (−10,10.2,9.8,10) are close enough to each other so that they are grouped together and are represented by the 2×2 red and green blocks on the diagonals of the corresponding inc-iVAT images in Figures 15(e) and 15(f). The structure of IRIS suggested by the image in Figure 12 is not visible in Figure 15 because images shown in Figure 15 are the zoomed right-bottom corners of the inc-iVAT images, not the entire inc-iVAT images.

Thresholds α and β clearly influence the success of inc-VAT/inc-iVAT. The sensitivity of inc-iVAT to these parameters is illustrated in Figure 16, where the anomaly detection experiment is repeated on $IRIS_{155}$ for different values of α and β. Figure 16(a) plots d_n against various thresholds for α shown by horizontal black lines. If $\alpha < 1$, there is a risk of false alarms (i.e., normal points being declared anomalous). In this experiment, false alarms begin when alpha drops below 0.44. This is why, we use a lower bound of 1 for α. The four zones shown in Figure 16(a) and the detection rates for each are as follows:

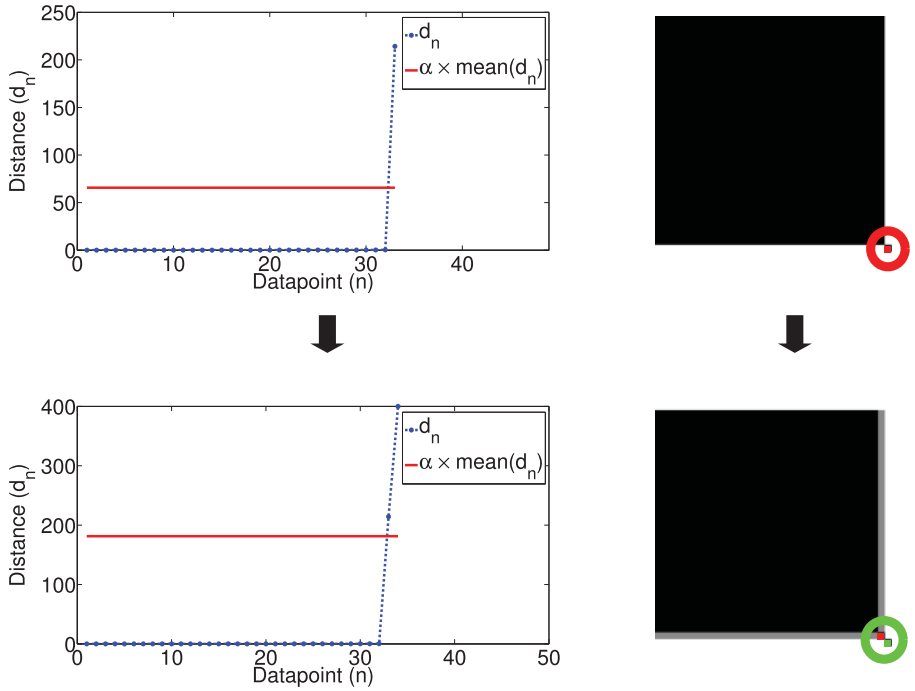


Fig. 14. d_n plot and inc-iVAT image for datapoints 34 (top) and 35 (bottom) on $IRIS_{155}$.

—Zone 1 : $\alpha \in [1, 19.09)$ five detects
—Zone 2 : $\alpha \in [19.09, 62.38)$ three detects
—Zone 3 : $\alpha \in [62.38, 69.68)$ one detect
—Zone 4 : $\alpha \in [69.68, \infty)$ no detect

For $1 \leq \alpha < 19.09$, inc-iVAT identifies all the anomalies, whereas for $19.09 \leq \alpha < 62.38$, inc-iVAT can identify all anomalies except $(10, 10, 10, 10)$ and $(10, 10.2, 9.8, 10)$ at locations 21 and 94, respectively, giving an accuracy of 60%. If we increase α further to lie in the range $62.38 \leq \alpha < 69.68$, inc-iVAT identifies only one anomaly $(-9.6, 10.2, -10.7, 8.9)$ at location 70. For $\alpha \geq 69.68$, none of the five anomalies can be detected.

The value of β determines the maximum number of datapoints in a distant cluster for it to be declared anomalous. Figure 16(b) shows the accuracy (i.e., the detection rate) of inc-iVAT in determining anomalies for $\alpha = 10$ as β varies from 0.001 to 1. For $\beta < 0.007$, inc-VAT identifies only the anomaly $(-9.6, 10.2, -10.7, 8.9)$ at location 70 as it is the only isolated anomaly and the other four are grouped as pairs. For $0.007 \leq \beta \leq 0.9$, inc-iVAT visually identifies all five anomalies. The video attached as supplementary material shows the inc-iVAT images of $IRIS_{155}$ dataset, detecting anomalies as and when they happen. The video can also be found at Internet of Things: A pilot deployment in the city of Melbourne, Australia [2015] (scroll to the bottom of the webpage).

9.2. Heron Island Weather Station Data Experiment

This experiment is performed on a real-life dataset collected from the Heron Island weather station deployed on the Great Barrier Reef, Australia [Bezdek et al. 2011]. The Heron Island data has three variables: (air) humidity, (air) pressure, and (air)

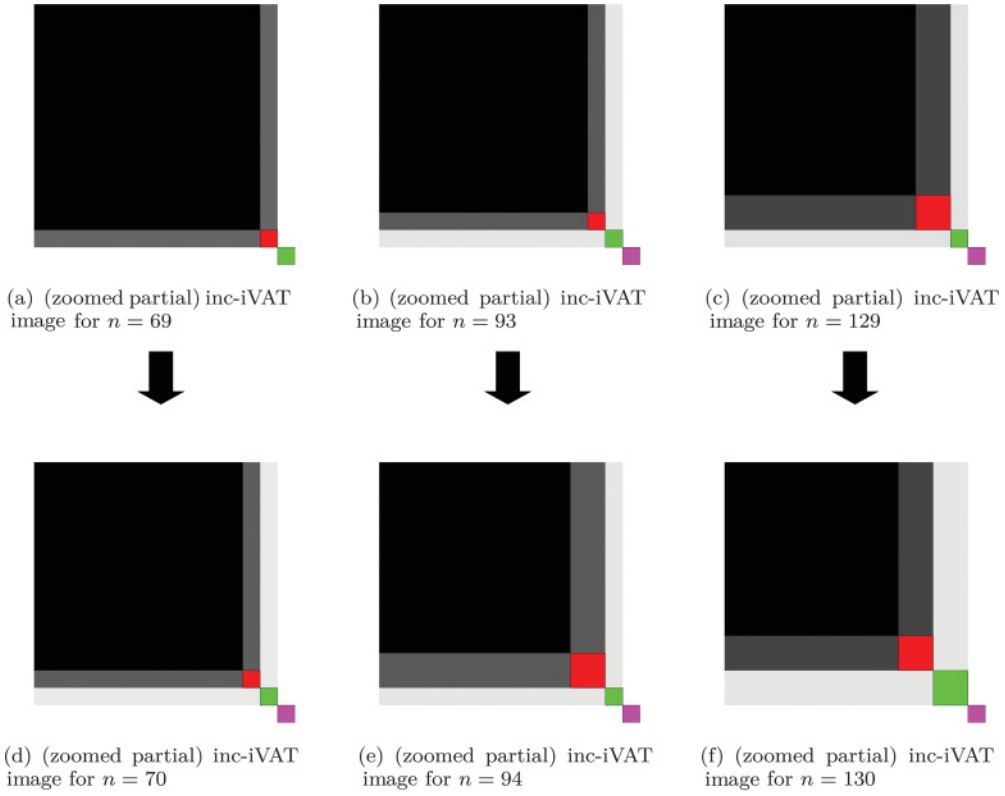
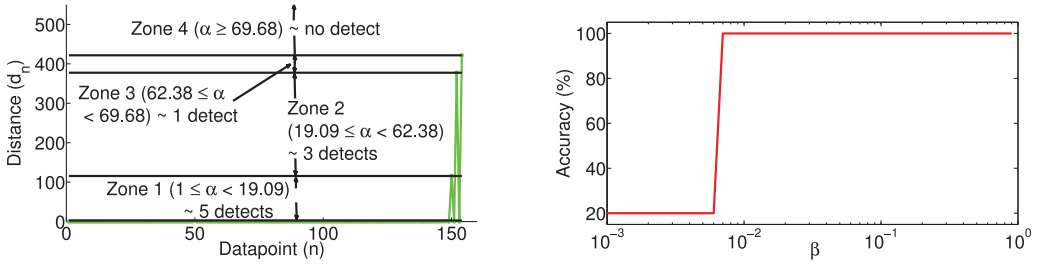


Fig. 15. inc-iVAT images before (top) and after (bottom) anomalies at locations 70, 94, and 130.

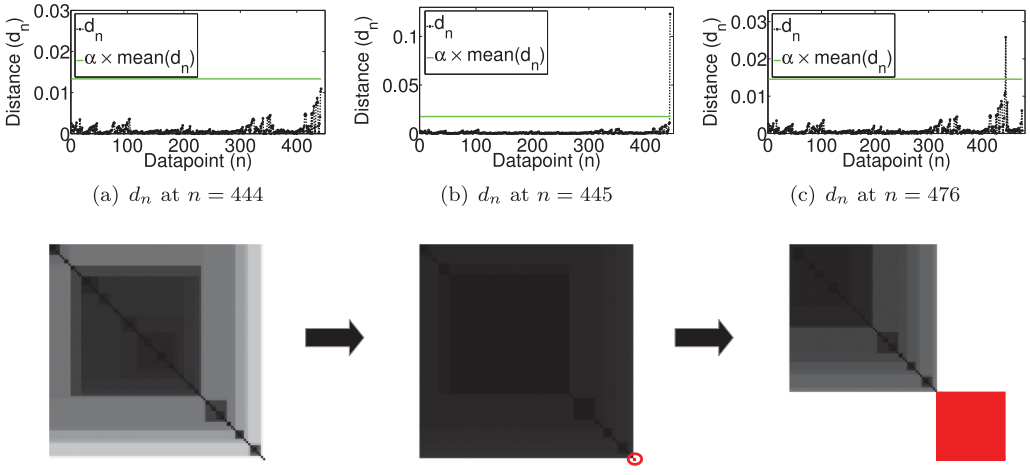


(a) Different values of α for $\beta = 0.02$ (b) Different values of β for $\alpha = 10$

Fig. 16. Experiment to show the effect of different values of α and β for anomaly detection on $IRIS_{155}$.

temperature. The data were collected every 10min from 9:00 am to 3:00 pm each day for the 30 days beginning 21-Feb-2009 and ending 22-Mar-2009. For this experiment, we have set the parameters $\alpha = 15$ and $\beta = 0.04$.

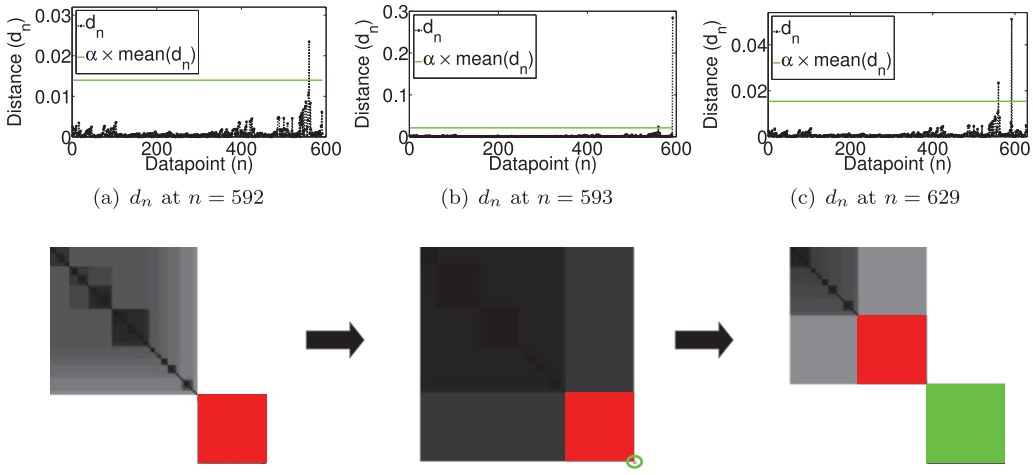
inc-VAT/inc-iVAT finds two (visual) anomalies in this data. The first anomaly occurs from datapoints 445 to 476 (5th March 9.00 AM to 2.10 PM) and the second one from datapoints 593 to 629 (9th March). Figures 17(a–c) plot the d_n just before the first anomaly ($n = 444$), just after the first anomalous point ($n = 445$), and after inclusion of all the 32 anomalous points ($n = 476$). The corresponding inc-iVAT images (zoomed right-bottom corner) are shown in Figures 17(d–f). It is hard to see the single red pixel



(a) d_n at $n = 444$ (b) d_n at $n = 445$ (c) d_n at $n = 476$

(d) (zoomed partial) inc-iVAT image at $n = 444$ (e) (zoomed partial) inc-iVAT image at $n = 445$ (f) (zoomed partial) inc-iVAT image at $n = 476$

Fig. 17. Heron Island data: d_n plot and inc-iVAT image for 5th March anomaly.



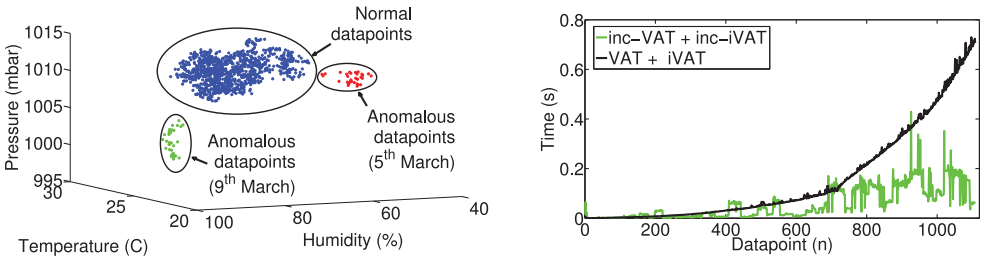
(a) d_n at $n = 592$ (b) d_n at $n = 593$ (c) d_n at $n = 629$

(d) (zoomed partial) inc-iVAT image at $n = 592$ (e) (zoomed partial) inc-iVAT image at $n = 593$ (f) (zoomed partial) inc-iVAT image at $n = 629$

Fig. 18. Heron Island data: d_n plot and inc-iVAT image for 9th March anomaly.

at the bottom-right corner in Figure 17(e), so we have circled it for emphasis. Similar figures for the second anomaly corresponding to 9th March data are shown in Figure 18. Figure 18 already contains a block of 32×32 red pixels corresponding to the first set of 32 anomalies, so there are two blocks of (one red and one green) pixels in Figure 18(f) after inc-iVAT has stepped through the second set of anomalies.

This Heron Island dataset was analyzed in Bezdek et al. [2011] using ellipsoidal chains for anomaly detection. Figures 6(d–f) in that paper show three *static* iVAT images of 2D projections of the humidity (H), pressure (P), and temperature (T) of the data in the coordinate pairs (H,P), (T,P), and (T,H). The (H,P) and (T,P) iVAT images in



(a) Normal datapoints (blue) and anomalies (red ($5th$ March) and green ($9th$ March))

(b) Time comparison for VAT+iVAT and inc-VAT+inc-iVAT algorithm

Fig. 19. 3D scatterplot of Heron Island weather data and time comparison for anomaly detection using inc-iVAT.

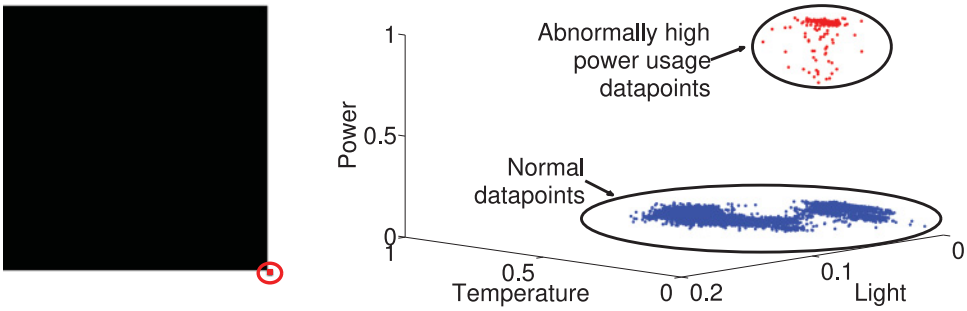
Figures 6(d) and 6(e) of Bezdek et al. [2011] show the 9th March anomaly corresponding to cyclone Hamish quite clearly. The (T,H) iVAT image at Figure 6(f) does identify the 5th March anomaly due to an unusual weather variation in T and H having regular P. However, when the chaining method is applied to the data in all three variables, only the 9th March anomaly is declared. This illustrates that inc-VAT/inc-iVAT provides a visual cue for both anomalous clusters directly from the 3D data, which are scatter plotted in Figure 19(a). The blue cluster in Figure 19(a) represents normal weather. The red cluster corresponds to the 5th March (T,H) anomaly and the green cluster are the measurements taken on the 9th March when the weather was adversely affected by cyclone Hamish. Figure 19(b) shows the time comparison for VAT + iVAT and inc-VAT + inc-iVAT for the Heron Island data. As the number of datapoints increases, VAT becomes computationally intensive, but inc-VAT takes only a fraction of the time taken by VAT. The video attached as supplementary material shows the 3D scatterplot of (air) humidity, (air) pressure, and (air) temperature, the MST cut magnitude (d_n), and the inc-iVAT image based on Euclidean distances in $\mathbb{R}^3$. The video clearly shows the anomalous event detection on 5th March and 9th March. The video can also be found at Internet of Things: A pilot deployment in the city of Melbourne, Australia [2015] (scroll to the bottom of the web page).

9.3. REDUCE Energy Data Experiment

This experiment is performed on the REDUCE energy dataset, which was collected from a real sensor network deployment at the Guildford campus in CCSR, the University of Surrey, United Kingdom, as part of the REDUCE project [Murtagh et al. 2013; REDUCE: Reshaping Energy Demands Using ICT 2013]. This project aims at monitoring energy usage of the devices and user context in an indoor office environment. The experimental setup consisted of 127 nodes deployed in two floors. Each sensor measured the total power, reactive power, phase angle, RMS voltage, RMS current, light, temperature, PIR (motion sensor), noise level, and vibration at each desk. For this anomaly detection experiment, we use seven of the twelve sensor nodes deployed in one of the rooms on the second floor, called Room A.

The sensors have node IDs 90, 91, 92, 94, 106, 108, and 109. The data are collected during a 24 hour period on 19th June 2012 at 10s sampling interval. This provides a dataset of 8,640 observations streaming in time for our example. The features used are temperature, light, and total power. This experiment uses the values of the parameters as $\alpha = 400$ and $\beta = 0.03$.

Figure 20(a) shows the inc-iVAT image for node 109. The anomaly appears as a single red pixel in the bottom-right corner of the image when first encountered in the streaming data. View (b) shows the 3D scatterplot of all the datapoints, where



(a) inc-iVAT image (b) Normal data (blue dots) and anomalies (red dots)

Fig. 20. Energy data: inc-iVAT image and 3D scatterplot of node 109 data.

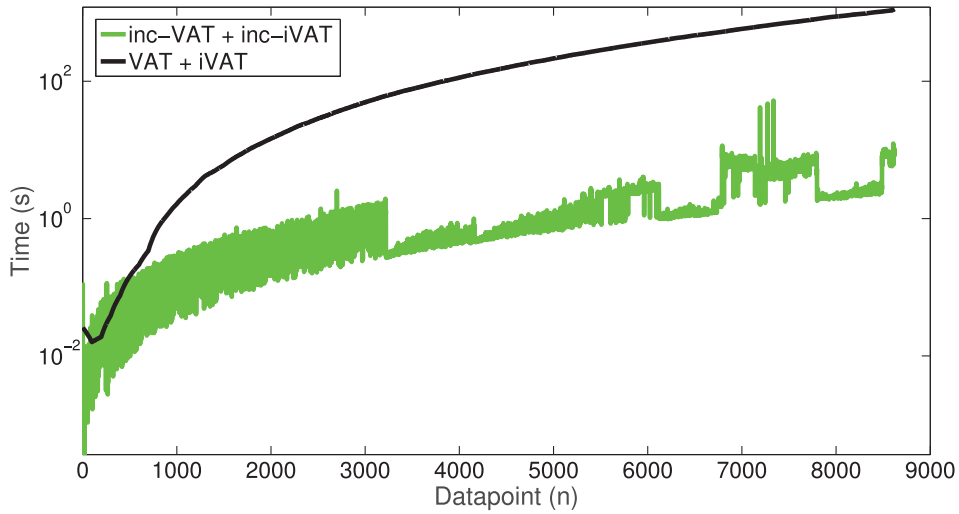


Fig. 21. Time comparison of VAT/iVAT and inc-VAT/inc-iVAT algorithms for the REDUCE energy dataset.

normal datapoints are shown by blue dots and anomalies representing abnormally high power usage are shown by red dots. The red pixel in Figure 20(a) grows into a red block in the inc-iVAT image as data continues to arrive so that the final inc-iVAT image looks very similar to Figure 17(f). Other nodes in the network did not exhibit this behavior. The results are consistent with the results obtained on the same dataset in Rajasegarar et al. [2014]. The comparison of run time of VAT + iVAT to get a new iVAT image vs. inc-VAT + inc-iVAT to update the previous inc-iVAT image at the arrival of each new datapoint is shown in Figure 21 (time is in log scale). As the number of datapoints increases, VAT/iVAT becomes computationally intensive, but inc-VAT/inc-iVAT takes only a fraction of the time taken by VAT/iVAT (up to three orders of magnitude less). Over the period of 8,640 observation points in this experiment, the average computational time to create a new iVAT image is 273.26s, whereas the average time required to update the inc-iVAT image using inc-VAT/inc-iVAT reduces by 170 times to 1.61s.

9.4. Melbourne IoT Data Experiment

As part of the "IoT for creating smart cities" project [Internet of Things: A pilot deployment in the city of Melbourne, Australia 2015], the ISSNIP group of the University of Melbourne, ARUP [ARUP 2015], and the *City of Melbourne council* (CoM) have done a pilot deployment of IoT networks, in the Melbourne city, for monitoring environmental parameters [Internet of Things: A pilot deployment in the city of Melbourne, Australia 2015; Shilton et al. 2015; Gubbi et al. 2013]. Environmental sensors, measuring light levels, humidity, and temperature, have been deployed at Fitzroy Gardens (five sensors) and the library at Docklands (four sensors). The data is collected at 10min intervals, and the live data is available from the open data platform of the CoM.

We use the light levels, humidity, and temperature data obtained from four sensors at the Docklands library for the sliding window based clustering experiment. The four sensors are nodes: 506, 509, 510, and 511. We resampled the dataset so that we have one measurement of temperature, humidity, and light from all four sensors every 30min. These measurements are concatenated to obtain a 12-dimensional feature vector (4 sensors $\times$ 3 measurements per sensor (temperature, humidity, and light)). In total, we have measurements for roughly 72 days (3,447 12-dimensional feature vectors). We perform a window-based clustering experiment on this dataset with a window size of 2 days (2 days $\times$ 24 hours per day $\times$ 2 samples per hour $=$ 96). So inc-VAT/inc-iVAT adds sample points to the MST for the first 96 samples and then applies one dec-VAT/dec-iVAT operation to remove the oldest datapoint and one inc-VAT/inc-iVAT operation to insert the new datapoint into the MST. At the last step, we use only dec-VAT/dec-iVAT to bring the number of datapoints up to 2.

As it is not possible to examine the 12-dimensional data visually, the video attached as supplementary material shows the plot of temperature, humidity, and light for each of the sensors, the MST cut magnitude (d_n), and the inc-iVAT/dec-iVAT image based on Euclidean distances in $\mathbb{R}^{12}$. The video shows a visual estimate of the number of clusters at each instant of data collection. We use the parameters $\alpha = 6$ and $\beta = 0.1$ to perform the clustering and anomaly detection task. In the video, the anomalies are shown by black dots and clusters are shown by different colors in the plot of temperature, humidity, and light for all the sensors. The video can also be found at Internet of Things: A pilot deployment in the city of Melbourne, Australia [2015] (scroll to the bottom of the web page). Figure 22 shows a snapshot of the sliding time window from 04 Jan 2015, 07:00:00 to 06 Jan 2015, 06:59:59. The inc-iVAT/dec-iVAT image in the lower right corner of Figure 22 indicates the presence of three clusters by three dark blocks along the diagonal. The MST cut magnitude d_n is shown in the top-right corner along with a red horizontal line showing the threshold value of $\alpha \times \text{mean}(d_n)$. Clusters whose size is less than $\lfloor \beta \times n \rfloor = \lfloor 0.1 \times 96 \rfloor = \lfloor 9.6 \rfloor = 9$ are declared as anomalous. The three normal clusters are shown by red, green, and blue dots, and the anomalies are shown by black dots in the temperature, humidity, and light 3D plot for nodes 506, 509, 510, and 511.

10. DISCUSSION AND CONCLUSION

We proposed and developed inc-VAT/inc-iVAT, an incremental version, and dec-VAT/dec-iVAT, a decremental version of VAT/iVAT, for online visual assessment of evolving cluster structure in streaming data. We have shown that the time complexity of inc-VAT/dec-VAT and inc-iVAT/dec-iVAT compares favorably to that of VAT and iVAT, respectively. We discussed two applications, anomaly detection and sliding window based visual assessment of clustering for time series data using inc-VAT/inc-iVAT and dec-VAT/dec-iVAT models for online visual assessment of evolving cluster structure in streaming data. We have demonstrated that the new models can provide visual

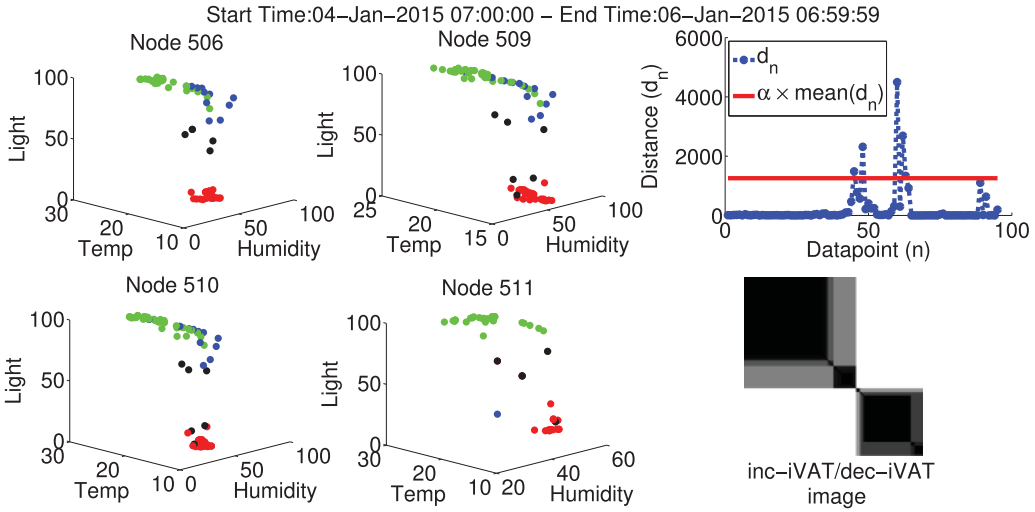


Fig. 22. A snapshot showing the three clusters (shown by red, green, and blue colors) and anomalies (shown by black) for sliding time window from 04 Jan 2015, 07:00:00 to 06 Jan 2015, 06:59:59.

cues to the onset of the new structure in streaming data using real-world datasets. The use of these models for anomaly detection and sliding time window based clustering is highly dependent on "right" choices for the parameters α and β. These two parameters can be manipulated to produce either excellent or horrible results. We have experimented with various combinations of α and β and conclude that a workable range for β is $0.01 - 0.10$. Recommending a useful range for α is much more challenging. In our examples, α ranges from $\alpha = 6$ for the Melbourne IoT data to $\alpha = 400$ for the REDUCE energy data. In the absence of a theoretical result for guidance, our best advice on this is the usual advice: Experiment with the type of data that your system observes until you find a combination of parameters that seems useful. In this way, you may develop some confidence that WYSIWYG (what you see is what you get). Finally, we point out that VAT (and hence, inc-VAT/dec-VAT) is limited by n, the number of datapoints that can be visualized before software and hardware constraints become problematic to store and visualize the reordered dissimilarity matrix. The practical limit depends on the system platform, but n cannot grow much beyond 5,000 before this problem stops the show. There is a scalable version of VAT, clusiVAT [Kumar et al. 2016], which intelligently samples the big data such the (relatively) small number of samples retain the approximate geometry of the big data before applying VAT/iVAT to the samples for cluster tendency assessment. In the future, we would like to extend the concept of incremental and decremental versions of VAT and iVAT to clusiVAT to alleviate the problem of hardware and software constraints of storing and visualizing reordered dissimilarity matrix as n increases.

REFERENCES

C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. 2003. A framework for clustering evolving data streams. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 81–92.

E. Anderson. 1935. The irises of the Gaspe peninsula. *Bulletin of American Iris Society* 59 (1935), 2–5.

P. Angelov. 2010. Evolving Takagi-Sugeno fuzzy systems from streaming data (eTS+). In *Evolving Intelligent Systems: Methodology and Applications*. IEEE Press.

ARUP. 2015. http://www.arup.com/Global_locations/Australia.aspx.

B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan. 2003. Maintaining variance and k-medians over data stream windows. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. 234–243.

J. L. Bentley and A. C. Yao. 1976. An almost optimal algorithm for unbounded searching. *Inform. Process. Lett.* 5, 3 (1976), 82–87.

J. Bezdek and R. Hathaway. 2002. VAT: A tool for visual assessment of (cluster) tendency. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)* (2002), 2225–2230.

J. C. Bezdek, R. J. Hathaway, and J. M. Huband. 2007. Visual assessment of clustering tendency for rectangular dissimilarity matrices. *IEEE Trans. Fuzzy Syst.* 15(5) (2007), 890–903.

J. C. Bezdek, S. Rajasegarar, M. Moshtaghi, C. Leckie, M. Palaniswami, and T. C. Havens. 2011. Anomaly detection in environmental monitoring networks. *Comput. Intell. Mag.* 6, 2 (2011), 52–58.

B. Delaunay. 1934. Sur la sphère vide. A la mémoire de Georges Voronoï. *Bulletin de l'Académie des Sciences de l'URSS, Classe des sciences mathématiques et naturelles* 6 (1934), 793–800.

E. D. Demaine, A. López-Ortiz, and J. I. Munro. 2000. Adaptive set intersections, unions, and differences. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 743–752.

E. W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numer. Math.* 1, 1 (1959), 269–271.

N. Elmqvist, P. Dragicevic, and J. D. Fekete. 2008. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Trans. Vis. Comput. Graphics* 14(6) (2008), 1539–1548.

V. Estivill-Castro and D. Wood. 1992. A survey of adaptive sorting algorithms. *ACM Comput. Surv.* 24, 4 (1992), 441–476.

D. Greene, D. Archambault, V. Belák, and P. Cunningham. 2015. TextLuas: Tracking and visualizing document and term clusters in dynamic text data. *CoRR* abs/1502.04609 (2015).

J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. 2013. Internet of things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* 29(7) (2013), 1645–1660.

R. Hathaway, J. Bezdek, and J. Huband. 2006. Scalable visual assessment of cluster tendency for large data sets. *Pattern Recognit.* 39 (2006), 1315–1324.

T. Havens and J. Bezdek. 2012. An efficient formulation of the improved visual assessment of cluster tendency (iVAT) algorithm. *IEEE Trans. Knowl. Data Eng.* 24(5) (2012), 813–822.

T. Havens, J. Bezdek, J. Keller, M. Popescu, and J. Huband. 2009. Is VAT really single linkage in disguise? *Ann. Math. Artif. Intell.* 55(3–4) (2009), 237–251.

Z. He, X. Xu, and S. Deng. 2003. Discovering cluster-based local outliers. *Pattern Recognit. Lett.* 24, 910 (2003), 1641–1650.

Internet of Things: A pilot deployment in the city of Melbourne, Australia. 2015. Retrieved 15 Oct 2016 from http://issnip.unimelb.edu.au/research_program/Internet_of_Things/iot_deployment.

V. Jarník. 1930. O jistém problému minimálním. *Práce Moravské Přírodovědecké Společnosti* 6 (1930), 57–63.

D. G. Kirkpatrick and R. Seidel. 1986. The ultimate planar convex hull algorithm. *SIAM J. Comput.* 15, 1 (1986), 287–299.

J. Kruskal. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proceedings of the American Mathematical Society, 7*.

D. Kumar, J. C. Bezdek, M. Palaniswami, S. Rajasegarar, C. Leckie, and T. C. Havens. 2016. A hybrid approach to clustering in big data. *IEEE Trans. Cybern.* 46, 10 (2016), 2372–2385.

D. Kumar, J. C. Bezdek, S. Rajasegarar, C. Leckie, and M. Palaniswami. 2015. A visual-numeric approach to clustering and anomaly detection for trajectory data. *Vis. Comput.* (2015), 1–17.

D. Kumar, M. Palaniswami, S. Rajasegarar, C. Leckie, J. Bezdek, and T. Havens. 2013. clusiVAT: A mixed visual/numerical clustering algorithm for big data. In *Proceedings of the IEEE International Conference on Big Data*. 112–117.

O. D. Lampe and H. Hauser. 2011. Interactive visualization of streaming data with kernel density estimation. In *Proceedings of the IEEE Pacific Visualization Symposium (PACIFICVIS)*. 171–178.

W. B. March, P. Ram, and A. G. Gray. 2010. Fast Euclidean minimum spanning tree: Algorithm, analysis, and applications. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 603–612.

M. Moshtaghi, J. C. Bezdek, C. Leckie, S. Karunasekera, and M. Palaniswami. 2015. Evolving fuzzy rules for anomaly detection in data streams. *IEEE Trans. Fuzzy Syst.* 23, 3 (2015), 688–700.

N. Murtagh, M. Nati, W. R. Headley, B. Gatersleben, A. Gluhak, M. A. Imran, and D. Uzzell. 2013. Individual energy use and feedback in an office setting: A field trial. *Energy Policy* 62 (2013), 717–728.

S. Pettie and V. Ramachandran. 2002. An optimal minimum spanning tree algorithm. *J. ACM* 49, 1 (2002), 16–34.

F. P. Preparata and M. I. Shamos. 1985. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY.

R. Prim. 1957. Shortest connection networks and some generalizations. *Bell Syst. Techn. J.* 36, 6 (1957), 1389–1401.

S. Rajasegarar, A. Gluhak, M. A. Imran, M. Nati, M. Moshtaghi, C. Leckie, and M. Palaniswami. 2014. Ellipsoidal neighbourhood outlier factor for distributed anomaly detection in resource constrained networks. *Pattern Recognit.* 47, 9 (2014), 2867–2879.

S. Rajasegarar, C. Leckie, M. Palaniswami, and J. Bezdek. 2006. Distributed anomaly detection in wireless sensor networks. In *Proceedings of the IEEE International Conference on Communication Systems (ICCS)* (2006), 1–5.

S. Rajasegarar, M. Palaniswami, C. Leckie, and J. Bezdek. 2007. Quarter sphere based distributed anomaly detection in wireless sensor networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*. 3864–3869.

K. Reda, C. Tantipathananandh, A. Johnson, J. Leigh, and T. Berger-Wolf. 2011. Visualizing the evolution of community structures in dynamic social networks. In *Proceedings of the Eurographics/IEEE - VGTC Conference on Visualization (EuroVis)*. 1061–1070.

REDUCE: Reshaping Energy Demands Using ICT. 2013. Retrieved 15 Oct 2016 from http://info.ee.surrey.ac.uk/CCSR/REDUCE/.

M. I. Shamos and D. Hoey. 1975. Closest-point problems. In *Proceedings of the Annual Symposium on Foundations of Computer Science (SFCS)*. 151–162.

A. B. Sharma, L. Golubchik, and R. Govindan. 2010. Sensor faults: Detection methods and prevalence in real-world datasets. *ACM Trans. Sensor Netw.* 6, 3, Article 23 (2010), 23:1–23:39 pages.

A. Shilton, S. Rajasegarar, C. Leckie, and M. Palaniswami. 2015. DP1SVM: A dynamic planar one-class support vector machine for internet of things environment. In *Proceedings of the International Conference on Recent Advances in Internet of Things (RIoT)*. 1–6.

A. Tartakovsky and V. Veeravalli. 2008. Asymptotically optimal quickest change detection in distributed sensor systems. *Sequential Anal.* 27 (2008), 441–475.

L. Wang, X. Geng, J. Bezdek, C. Leckie, and K. Ramamohanarao. 2010. Enhanced visual analysis for cluster tendency assessment and data partitioning. *IEEE Trans. Knowl. Data Eng.* 22(10) (2010), 1401–1414.

P. C. Wong, H. Foote, D. Adams, W. Cowley, and J. Thomas. 2003. Dynamic visualization of transient data streams. In *Proceedings of the Annual IEEE Conference on Information Visualization (INFOVIS)*. 97–104.

XmdvTool. 2015. Retrieved 15 Oct 2016 from http://davis.wpi.edu/xmdv/.

D. Yang, E. A. Rundensteiner, and M. O. Ward. 2012. Shared execution strategy for neighbor-based pattern mining requests over streaming windows. *ACM Trans. Database Syst.* 37, 1, Article 5 (2012), 5:1–5:44 pages.

D. Yang, E. A. Rundensteiner, and M. O. Ward. 2013a. Mining neighbor-based patterns in data streams. *Inf. Syst.* 38, 3 (2013), 331–350.

D. Yang, K. Zhao, H. Lu, M. Hasan, E. A. Rundensteiner, and M. O. Ward. 2013b. Mining and linking patterns across live data streams and stream archives. *Proc. VLDB Endowment* 6, 12 (2013), 1346–1349.

Y. Yao, A. B. Sharma, L. Golubchik, and R. Govindan. 2010. Online anomaly detection for sensor systems: A simple and efficient approach. *Perform. Eval.* 67, 11 (2010), 1059–1075.

A. Zhou, F. Cao, W. Qian, and C. Jin. 2008. Tracking clusters in evolving data streams over sliding windows. *Knowl. Inf. Syst.* 15, 2 (2008), 181–214.

Received June 2015; revised May 2016; accepted September 2016