Received 16 November 2022, accepted 2 December 2022, date of publication 12 December 2022, date of current version 19 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3228238

RESEARCH ARTICLE

Piecemeal Clustering: A Self-Driven Data Clustering Algorithm

MD. MONJUR UL HASAN[1], REZA SHAHIDI[1], (Senior Member, IEEE), DENNIS K. PETERS[1], (Senior Member, IEEE), LESLEY JAMES[2], AND RAY GOSINE[1]

[1]Department of Electrical and Computer Engineering, Memorial University of Newfoundland, St. John's, NL A1C 5S7, Canada
[2]Department of Process Engineering, Memorial University of Newfoundland, St. John's, NL A1B 3×5, Canada

Corresponding author: Md. Monjur Ul Hasan (mmuhasan@mun.ca)

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), in part by Equinor (formerly Statoil), and in part by Memorial University.

ABSTRACT Various approaches have been discussed in the literature for the clustering of data, such as partitioning, hierarchical, and machine learning methods. Most of the approaches require some prior knowledge about the clusters, such as their total number. Furthermore, some previous algorithms are not robust enough to process higher-dimensional data or require a large amount of memory for computations. We propose, herein, a data clustering algorithm, **Piecemeal Clustering**, that successfully clusters data without prior knowledge of the number of clusters. The proposed clustering algorithm uses the similarity and density of the data to identify the number of clusters in the data set and works with both low- and high-dimensional data. We demonstrate the power of the proposed **Piecemeal Clustering** algorithm with two real-world data sets. It is found that the proposed algorithm outperforms seven other state-of-the-art algorithms on both of these data sets.

INDEX TERMS Data clustering, agglomerative clustering, density-based clustering, unsupervised learning.

NOMENCLATURE

Δ_{ij}	Distance of a data point from cluster center.
CD	Distance matrix (Combined).
CD_d	Distance matrix (Euclidean).
CD_s	Similarity matrix (Cosine).
cL	Learning rate in current iteration.
cR_D	Distance radius in current iteration (Euclidean).
cR_S	Similarity radius in current iteration (Cosine).
D_T	Distance cutoff threshold (Euclidean).
$E(i,j)$	Minimum combined distance of a data point D_i in cluster Cl_i to another point D_j in cluster Cl_j.
$E_d(i,j)$	Minimum Euclidean distance of a data point D_i in cluster Cl_i to another point D_j in cluster Cl_j.
$E_s(i,j)$	Minimum cosine similarity of a data point D_i in cluster Cl_i with another point D_j in cluster Cl_j.
I	Combined Influence.
I_s	Similarity Influence.
I_d	Distance Influence.
K	Total number of clusters determined by the algorithm.
L	Learning rate.
M	Number of clusters generated after the Pre-Clustering phase.
N	Number of iterations in Training phase.
n	Total number of data points in the data set.
R_D	Distance radius for Training (Euclidean).
R_S	Similarity radius for Training (Cosine).
S_T	Similarity cutoff threshold (Cosine).
T	Cutoff threshold.
T_{DS}	Combined cutoff threshold.
V_N	A calculated difference factor between Euclidean and cosine distance matrices.

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

I. INTRODUCTION

Data clustering, or simply *clustering*, refers to the organization of data into different groups where data in the same group are similar, while data in different groups are dissimilar [1]. Currently, it is possible to collect enormous amounts of data and store them. By using understandable and meaningful data clustering methods, better decisions can be made using these data. Some real-world application areas of such clustering methods include: weather forecasting [2], image processing [3], and general big-data analysis [4].

Data may be easily clustered if a small set of rules can be applied to determine membership in any cluster, which is rarely possible with real world data. A good clustering algorithm should possess all of the following characteristics: 1) automatically identifies the total number of clusters from the data 2) is efficient and scalable to work with large-scale high-dimensional data, 3) considers all dimensions of the data in the clustering process, and 4) is not sensitive to outliers and noise. A number of clustering algorithms targeting a variety of scenarios has been developed and studied [5], [6], [7], [8], [9].

Some existing clustering algorithms are able to produce a pre-defined number K of good quality clusters from high-dimensional large data that do not include outliers or noise, e.g., *K-means* [10], *K-medioids* [11], *PAM* [12], and *CLARANS* [13]. Other algorithms, including *GMM* [14], and Self-Organizing Maps (*SOM*) [15] are not sensitive to outliers or noise, and can produce good-quality clusters from high-dimensional large data. Their performance, however, depends heavily on the appropriate selection of underlying models and model parameters. *SOM* is widely used among the algorithms of this class [16], [17], [18], [19]. There are density-based clustering algorithms (e.g., *DBSCAN* [20]) which perform clustering in real hyper-space with the number of clusters generated from the data without any prior knowledge. A variety of approaches can overcome the limitations of these algorithms, such as trial and error to find the right set of parameters, removing noise from the data before clustering, or using a combination of multiple methods. Further discussion of clustering algorithms and their limitations, as well as of recent studies, is given in Section II.

Despite being an active study area, challenges in the design and development of data clustering algorithms remain open. In addition, lower-cost data collection and storage have continuously been enabling many new domains to store enormous amounts of data. As a result, more domain experts and practitioners have been frequently trying to use data clustering algorithms. Therefore, there has been a gradual shift in the required capabilities of clustering algorithms. No single algorithm fully addresses all of the challenges [1], [21].

To resolve the above-mentioned problems, we propose a robust and generic data clustering algorithm. The algorithm can calculate the number of clusters automatically. At the same time, it can handle both lower- and higher-dimensional data. The algorithm uses all data dimensions for the clustering, yet is scalable to larger data sets. The final result of the data clustering is also not greatly influenced by noise present in the data.

The proposed algorithm can help in different scientific studies and solve practical problems. For example, lithofacies identification is an important study for understanding the geology of a location. After obtaining the seismic images, geologists often bore exploratory wells for further study. They often collect wireline logs from such wells, e.g., Gamma Rays, Resistivity, Resistivity, and Porosity. Such logs are then carefully analyzed with manual effort to mark different lithofacies. This is clearly a data clustering problem where the number of clusters is unknown prior to the clustering and the sizes and shapes of the clusters are irregular. The data point density in the hyper-space is also not regular and therefore a good number of data points can be found as noise. Density based data clustering algorithms available in the literature may be able to cluster such data sets and identify the number of clusters in the data set, but they filter out many data points as noise. Therefore, only supervised and semi-supervised classifiers are used to solve the problem [22], which does not eliminate the manual process. The proposed algorithm may solve this problem since it can identify the number of clusters on its own, and also works with noisy data. In Section IV and V we demonstrate the novelty of the proposed algorithm with further details.

The proposed algorithm utilizes a density-based clustering method combining the concepts of hierarchical clustering [23], model-based unsupervised learning [24], and density based data clustering. The algorithm uses two vector distance measures: Euclidean distance and cosine similarity, to measure pairwise distance and similarity, respectively. While both of the measures make use of all the data dimensions, the calculations are relatively inexpensive. Therefore, the algorithm is not limited by data dimensionality and is capable of handling larger high-dimensional data.

The algorithm calculates pairwise distance and similarity values and uses their distributions in real hyper-space to identify the clusters in three phases. First, the algorithm starts with an agglomerative hierarchical clustering approach and produces a large number of clusters, where each cluster centroid is the representative model of its cluster. In the next phase, a model-based clustering approach is used to adjust the cluster centers to form clusters of cluster centers. A new unsupervised learning method is proposed for this phase. Clusters of cluster centers are then merged in the third and final phase of the algorithm. These phases of the proposed algorithm are named Pre-Clustering, Training, and Post-Processing, respectively. Since the clusters are formed progressively through the three phases, we call the algorithm **Piecemeal Clustering**.

This algorithm has three parameters: 1) the cutoff threshold, T, 2) the learning rate, L, and 3) the number of iterations, N. The cutoff threshold, T, is conventionally used in agglomerative hierarchical clustering and carries the same meaning and purpose in this algorithm, and is only used in the Pre-Clustering phase. The learning rate, L, and number of iterations, N, are also conventional parameters for machine learning training. These two parameters are used in the

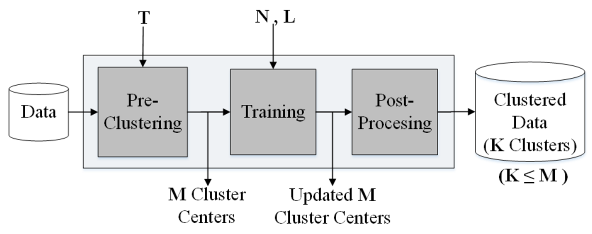


FIGURE 1. Block diagram showing the three phases of the Piecemeal clustering algorithm. The diagram also outlines the input parameters of each of the phases and their respective outputs.

Training phase, and carry the same meaning and purpose. The last phase, Post-Processing, is parameter-less, and produces the final clustering result. Fig 1 shows a flowchart for the proposed algorithm.

We test the proposed algorithm on two real world data sets and compare the results with seven other well-known clustering algorithms in the literature. In contrast to the algorithms in the literature, **Piecemeal Clustering** is able to correctly and independently identify the total number of natural clusters. **Piecemeal Clustering** is also able to correctly map the highest number of data points to their respective clusters among the tested algorithms.

The rest of the paper is organized as follows: Section II discusses related work in the literature. Section III explains the proposed algorithm, and Section IV presents experimental results from the algorithm on both data sets. Section V compares the proposed algorithm to the other algorithms tested. Finally, Section VI presents conclusions.

II. LITERATURE REVIEW

The data clustering literature is vast, covering different aspects of the same general idea. While it is difficult to comprehensively discuss the prior literature in full depth, the following discussion categorizes clustering algorithms based on partitioning, hierarchy, distance and density, and underlying models.

Clustering via partitioning of the data set is the most common technique. Methods such as *k-means*, *k-medioids*, *PAM*, and *CLARANS* are algorithms in this group. The core idea of these algorithms is to start with a known number of clusters, *k*, fed in as input and then partition the data sets into *k* segments. Each of the segments is considered to be a cluster. A detailed review of this group of clustering algorithms can be found in [1]. Because the number of clusters is preset, the results from these algorithms strongly depend on the number as well as the shapes and the sizes, of the clusters. Outliers also significantly disrupt the quality of the clusters.

Another approach to identifying the clusters in a data set is to construct hierarchical relationships among the data [25]. Two approaches are used for this type of clustering: agglomerative and divisive. In the agglomerative approach, each data point is considered to be a cluster at the beginning; after which, the two clusters with the nearest centers are merged into a new cluster, continuing until only one cluster is left. Many variations of this core idea exist in the literature, including use of underlying models [26], neighbor graphs [27], centroids of trees [28], or shared subordinates [29]. In the divisive approach, all data points are initially considered to be part of a single cluster and the algorithm recursively splits the initial cluster into multiple clusters [25]. While this group of clustering algorithms is suitable for data sets having arbitrary attributes and shapes, such clustering algorithms require a large amount of memory to execute, and are therefore not suitable for large data sets. They also often produce sub-optimal solutions.

A major disadvantage of the above-mentioned types of clustering algorithms is that formal inference is not possible, since the results of the algorithms are based on heuristics. In contrast, model-based clustering algorithms generally require analysts to formulate the probabilistic model explicitly and then fit the data to the formulating model using machine learning. Two main types of learning methods are used for this type of clustering: statistical learning and neural network learning [30]. *COBWEB* and *GMM* are statistical-learning based clustering methods, while *SOM* is a widely-accepted neural-network clustering algorithm.

Among the neural-network based data clustering methods, unsupervised learning algorithms [31] allow for the determination of a natural grouping of data by looking at the structures within the data without the grouping being tied to a specific outcome. *SOM* is a widely-used unsupervised learning based clustering algorithm. A general disadvantage of model-based clustering algorithms is that prior knowledge of the data is required to find a good model and the result is sensitive to the choice of parameters.

Another approach to data clustering is the use of the local density and shortest distance of the data in hyper-space. *DBSCAN* is the first algorithm proposed of this kind. One unique advantage of such algorithms is that they do not require prior knowledge of the number of clusters in the data set. The algorithms themselves can identify the number of clusters based on the densities of the data points in their respective spaces. Another unique feature of these algorithms is that they can identify outliers in the data and separate them. A key disadvantage of the original *DBSCAN* algorithm is that it does not work well with input data of varying density [32]. A wide variety of algorithms based on *DBSCAN* has since been proposed, *DBCLASD* [33], *ST-DBSCAN* [34] and *HDBSCAN* [35]. These variants include improvements such as: better cluster quality and higher algorithmic efficiency, and reducing the number of parameters [36]. Interesting combinations of different concepts have also been proposed: *RNN-DBSCAN* [37] incorporates *k*-neighbourhood graph traversal with *DBSCAN* to handle large variations in cluster densities; *HDBSCAN** [38] is an improvement derived from *HDBSCAN*, which combines the concept of hierarchical clustering with density-based clustering. *Block-DBSCAN* [39], one of the most recent variants, tries to

TABLE 1. Detailed comparison of the algorithms.

Algorithm	Time Complexity	Scalability	For Large Scale Data set	For High-Dimensional Data Set	Sensitivity to the Data Sequence	Sensitivity to Noise	Number of Parameters	Pre-Set Cluster Number	Reference
K-means	$O(knt)$	Medium	Yes	No	Highly	Highly	1	Yes	[10]
K-medioids	$O(k(n-k)^2)$	Low	No	No	Moderate	Little	1	Yes	[41]
PAM	$O(k^3n^2)$	Low	No	No	Moderate	Little	1	Yes	[40]
SOM	$O(nm)$	Low	No	Yes	Little	Little	2	No	[15]
DBSCAN	$O(n \log n)$	Medium	Yes	No	Moderate	Little	2	No	[20]
DBCLASD	$O(n \log n)$	Medium	Yes	Yes	Little	Little	0	No	[33]
HDBSCAN	$O(n^2)$	Medium	Yes	No	Moderate	No	1	No	[35], [42]
RNN-DBSCAN	$O(kn^2)$	Medium	No	Yes	Little	No	1	No	[37]
*HDBSCAN**	$O(n^2)$	High	Yes	Yes	Little	No	2	No	[38], [43]
Blocked-DBSCAN	$O(n^2)$	Low	Yes	Yes	Little	No	2	No	[39]

improve the algorithm's efficiency in handling large-scale, high-dimensional data sets.

A comparison of the clustering algorithms is shown below in Table 1. The table includes the algorithms that are related to the current study. A comprehensive comparison that covers a wide set of clustering algorithms can be found in [1], [21], and [40] .

III. PROPOSED ALGORITHM

The core idea of the proposed algorithm is to keep all data points that are *close* (within a certain Euclidean distance) and *similar* (within a certain bound of cosine similarity) to each other, in a single cluster. Determining the threshold for the data points to be close in the Euclidean space, and similar in the cosine similarity space, is challenging without prior knowledge of the data set. The **Piecemeal Algorithm** uses both of the spaces simultaneously with an adaptive approach to overcome this challenge.

A simple two-dimensional case of combining these two measures for grouping data points is shown in Fig 2. Four vectors are shown in each of Figs 2a and 2b. In Fig 2a, vectors **A** and **B** may be considered to be part of the same cluster. However, vector **C** may not be part of that cluster because **C** is close to neither **A** nor **B**, even though it is similar to **A**. A similar argument can be applied to vector **D**. On the other hand, in Fig 2b, vectors **C** and **D** may be part of the same cluster. However, vectors **A** and **B** may not be part of that cluster for the apparent reason of not being close to **C** or **D**. At the same time, while the distance between **A** and **B** is the same as the distance between **C** and **D**, **A** and **B** may not be part of the same cluster as they are not similar to one another.

Fig 2 shows a simple case that is easy to formulate. The distributions of data within a cluster and their complexity scale-up geometrically to higher-dimensional data. The following sections describe the three phases of the **Piecemeal Clustering** algorithm and explain how the clusters are formulated while handling the complexity of higher data dimensionality.

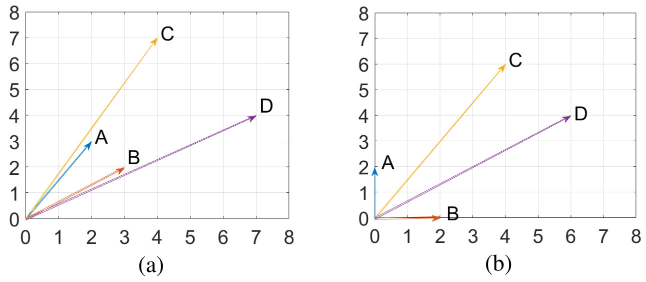


FIGURE 2. Example clustering of 2D data using Euclidean distances and cosine similarity. (a) A and C, having very similar orientation, cannot be part of the same cluster because of their distance. Same for B and D. (b) Euclidean distances between A and B, and C and D, are the same, however, all are not part of the same cluster.

A. PRE-CLUSTERING

The Pre-Clustering phase generates the initial set of clusters. It considers the local density and the shortest distances between data points, as in [44], but uses this concept within an agglomerative hierarchical framework to find clusters by applying a cutoff threshold. The objective is to generate an initial set of clusters for the next phase, where the cluster centers will be used as an initial model for the Training phase.

The Pre-Clustering phase uses a user-defined cutoff threshold parameter T ($0 \leq T \leq 1$), which is a fraction of the maximum Euclidean distance and cosine similarity between any two data points in the entire data set. The algorithm interprets this parameter as the maximum-allowed variation between members of any given cluster produced by this phase. Therefore, domain experts, using this algorithm, can incorporate their knowledge of unavoidable or natural variation within members of the same cluster. Intuitively, choosing a lower value for T will result in a larger number of smaller-sized clusters, while a higher value of T will result in a smaller number of larger-sized clusters. A value of $T = 0$ indicates there is no variation allowed in any of the clusters, leaving each data point as a cluster by itself. In contrast, $T = 1$ indicates the maximum distance and similarity between any two data points is the allowed degree

Algorithm 1 Pre-Clustering Algorithm

```
Input: data, $T$, $V_N$
Output: a list of cluster centers
1:  cluster_centers ← data
2:  while not_converge(cluster_centers, $T$) do
3:      $D_T$ ← dist_threshold(cluster_centers, $T$)
4:      $S_T$ ← ori_threshold(cluster_centers, $T$, $V_N$)
5:      $T_{DS} \leftarrow \sqrt{D_T^2 + S_T^2}$
6:      $CD$ ← pairwise_cluster_distances(
                  cluster_centers, $V_N$)
7:      for all $CD$ do
8:          $[i, j]$ ← find_next_nearest_pair($CD$)
9:          if is_nearby($i, j, T_{DS}$) then
10:             merge_clusters($i, j$)
11:         end if
12:         $CD(i, j) \leftarrow \infty$
13:     end for
14:     cluster_centers ←
                update_clusters(cluster_centers)
15: end while
```

of variation within a cluster, producing one cluster for the entire data set. The value of T can be calculated from the maximum Euclidean distance between the data points and knowledge about what is unavoidable or natural variation between members of each cluster.

Algorithm 1 shows the steps to produce clusters in this phase. The algorithm starts with identifying each of the data points as a separate cluster. As such, each data point becomes a cluster center (line 1). Then, the Pre-Clustering algorithm iteratively merges the clusters (line 2–13), reducing the total number of clusters at each iteration. This corresponds to an agglomerative approach.

The iterations stop when one of the following two conditions becomes true: (a) the last iteration fails to reduce the number of clusters, or (b) the total number of clusters is reduced to the reciprocal of T^2. Intuitively, when the first condition is reached, it indicates that no more merging is possible between any pair of clusters because the distance between them is greater than the cutoff threshold. For some data sets, this can be a case of a local minimum. The purpose of this phase is not to necessarily reach the global minimum but at the same time not to preclude the global minimum from future phases. This phase aims to create the initial set of cluster centers for the Training phase. After the Training phase, the goal of the Post-Processing phase is to merge the clusters to reach the global minimum.

The choice of a larger value for T runs the risk of over-clustering in this phase. The second condition is in place to avoid such a scenario, halting the iterative process, once a significantly-lower number of clusters, as calculated from the value of T, has been reached. This cutoff is important for the later phases of the algorithm which can only merge, and therefore only reduce the number of clusters. Over-clustering in this phase may result in an inaccurate clustering in later phases.

In each iteration, two threshold values: the *distance threshold*, D_T, and the *similarity threshold*, S_T, are calculated from all of the cluster centers. D_T is the fraction of the normalized maximum Euclidean distances between any two cluster centers calculated using (1).

$$D_T = \left(1 - \frac{\min_{i,j,i\neq j} ||\mathbf{C_i} - \mathbf{C_j}||}{\max_{i,j,i\neq j} ||\mathbf{C_i} - \mathbf{C_j}||}\right) \cdot T \tag{1}$$

where $||\cdot||$ denotes the Euclidean norm and $\mathbf{C_i}$ corresponds to the center of cluster i. Line 3 in Algorithm 1 implements this equation for D_T as the function *dist_threshold*(**cluster_centers**, T).

The value of S_T is calculated at line 4 within the function *ori_threshold*(**cluster_centers**, T, V_N), which uses (2). The equation is similar to that used for D_T, except cosine similarity is used in place of Euclidean distance, and the value is scaled by V_N, a value inversely-proportional to the mean Euclidean norm of all data points. A small positive offset is added to the mean Euclidean norm to ensure V_N is finite. Both D_T and S_T are combined to form T_{DS} in the next step (Line 5) to produce a single threshold. This value of T_{DS} helps to determine whether or not a pair of clusters can be merged. Therefore, the scaling of cosine similarity using V_N is important. Multiplication by the V_N is a mapping to account for differences between the Euclidean and cosine distance metrics. If the mean Euclidean distance of the data is smaller, then the data points will be closer to the origin. In such cases, data points within the same Euclidean distance will tend to have a larger difference in angle. As a result, larger threshold values should be used to separate different clusters based on cosine similarity, and vice versa.

$$S_T = \left(1 - \frac{\min_{i,j,i\neq j} \frac{\mathbf{C_i}\cdot\mathbf{C_j}}{||\mathbf{C_i}||||\mathbf{C_j}||}}{\max_{i,j,i\neq j} \frac{\mathbf{C_i}\cdot\mathbf{C_j}}{||\mathbf{C_i}||||\mathbf{C_j}||}}\right) \cdot T \cdot V_N \tag{2}$$

An array of pairwise cluster distances, **CD**, is then calculated (line 6), where $\mathbf{CD}(i, j)$ indicates the distance between the centers of clusters i and j. The value of $\mathbf{CD}(i, j)$ is calculated using the normalized distance $\mathbf{CD_d}(i, j)$ and the normalized similarity $\mathbf{CD_s}(i, j)$. The normalized distance $\mathbf{CD_d}(i, j)$ is the Euclidean distance between the centers of clusters i and j divided by the maximum Euclidean distance between any two cluster centers at that iteration. The normalized similarity $\mathbf{CD_s}(i, j)$ is also calculated in a similar fashion, except the cosine similarity is used in place of Euclidean distance. These two values are combined using (3) to generate $\mathbf{CD}(i, j)$.

$$\mathbf{CD}(i, j) = \sqrt{\mathbf{CD_d}^2(i, j) + V_N \cdot \mathbf{CD_s}^2(i, j)} \tag{3}$$

Then all pairs of clusters are considered for merging sequentially in non-decreasing order of their corresponding **CD** values (line 7–13). A pair of clusters (i, j) is merged if and only if both of the following two conditions hold:

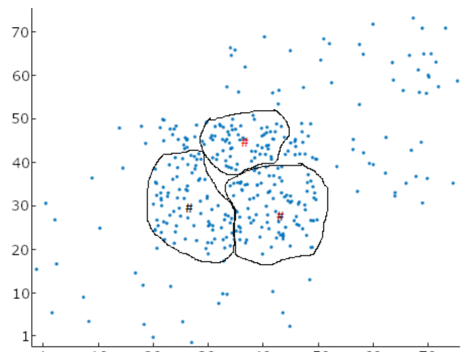


FIGURE 3. Example outcome of the Pre-Clustering phase on two-dimensional data. The figure shows a group of two dimensional points that are close to one another. The Pre-Clustering phase groups the points into three clusters. The '#' symbols indicates the calculated cluster centers' positions in each of the three clusters.

1) $\mathbf{CD}(i, j) < T_{DS}$
2) $\mathbf{CD}(m, n) < T_{DS}\ \forall m \ni$ clusters i and m have already been merged in this iteration, and $\forall n \ni$ clusters j and n have already been merged in this iteration.

These conditions are checked inside the function $is\_nearby(i, j, T_{DS})$ at line 9 of Algorithm 1 and the clusters are merged at line 10. During merging, all members of cluster j are marked as members of cluster i and the distance between the centroids of clusters i and j is set to ∞ so that the two clusters will not be considered for merging again in future iterations (line 12). This process is repeated, until all **CD** values have been processed.

Throughout this entire procedure, clusters are grouped together and new larger clusters created. Once all cluster pairs have been checked, the *cluster_centers* are updated to be the set of centroids of the newly-updated clusters (line 14). The process (line 2–15) is repeated with the updated *cluster_centers*, until the convergence criteria are satisfied at line 2.

The purpose of the Pre-Clustering phase is to create a basis for identifying the number of clusters from the data. The set of cluster centers generated in this phase is used as the initial set of models for the Training phase, described in the next subsection. At the end of training, the clusters are merged to form the appropriate clusters in Post-Processing. As a result, a small variation in T does not overly affect the final result. Fig 3 shows an example with two-dimensional data where three irregular clusters are formed within one apparent cluster because of the choice of a lower value for T. The algorithm's later phases can merge them to form an appropriate cluster. We recommend selecting a lower value for T to obtain a more granular set of clusters at the beginning. The user can later vary the initial value if they are unsatisfied with the result or want to try different scenarios.

B. TRAINING

The Training phase is also iterative, with cluster centers dynamically updated at each iteration. The basic steps for this phase are slightly modified from the steps of the training algorithm used for the SOM algorithm. This phase uses a user-defined number of iterations N, and learning rate L.

Algorithm 2 Training Algorithm

```
Input: data, cluster_centers, L, N
Output: Updated cluster centers
 1: $R_D \leftarrow \frac{\max_{i,j,i\neq j}\|\mathbf{C_i}-\mathbf{C_j}\| - \min_{i,j,i\neq j}\|\mathbf{C_i}-\mathbf{C_j}\|}{2}$
 2: $R_S \leftarrow \frac{\max_{i,j,i\neq j}\frac{\mathbf{C_i}.\mathbf{C_j}}{\|\mathbf{C_i}\|\|\mathbf{C_j}\|} - \min_{i,j,i\neq j}\frac{\mathbf{C_i}.\mathbf{C_j}}{\|\mathbf{C_i}\|\|\mathbf{C_j}\|}}{2}$
 3: for $iter \leftarrow 1, N$ do
 4:     $cR_D \leftarrow R_D \cdot e^{-\frac{iter}{L}}$
 5:     $cR_S \leftarrow R_S \cdot e^{-\frac{iter}{L}}$
 6:     $cL \leftarrow L \cdot e^{-\frac{iter}{L}}$
 7:     for all data do
 8:         cData $\leftarrow$ $get\_next$(data)
 9:         for all cluster_centers do
10:             cCluster $\leftarrow$
                    $get\_next$(cluster_centers)
11:             $d \leftarrow \|\mathbf{cData} - \mathbf{cCluster}\|$
12:             $s \leftarrow \frac{\mathbf{cData}\cdot\mathbf{cCluster}}{\|\mathbf{cData}\|\|\mathbf{cCluster}\|}$
13:             if $d < cR_D$ and $s < cR_S$ then
14:                 $I_D \leftarrow e^{-\frac{d^2}{2cR_D^2}}$
15:                 $I_S \leftarrow e^{-\frac{s^2}{2cR_S^2}}$
16:                 $I \leftarrow \sqrt{I_D \cdot I_S}$
17:                 cCluster $\leftarrow$ cCluster$-$
                        $cL \cdot I \cdot$ (cCluster $-$ cData)
18:                 $update\_cluster$(cluster_centres,
                        cCluster)
19:             end if
20:         end for
21:     end for
22: end for
```

Algorithm 2 shows the detailed steps of this phase. The algorithm uses exponentially-shrinking radii values. The radii values start with R_D and R_S for distance and similarity, respectively, calculated using the equations shown in lines 1 and 2 of Algorithm 2, where $\mathbf{C_i}$ indicates the center of cluster i. At each training iteration R_D, R_S, and L decay exponentially, and new values are calculated (lines 4–6). After calculating the new values, each of the data points is taken one at a time, in random order (line 8). For each of the data points, all the cluster centers, which are within the decayed distances and similarity values for that iteration, have their positions updated (lines 9–20). The degree of update is determined by the distances and similarities between each of the data points and the cluster centers. The closer and more similar a data point is to a cluster center, the stronger the position update that the cluster center receives. This is achieved by calculating the influence factors for both of the metrics separately (lines 14–15). The influence decreases exponentially with increasing distance and similarity. Both of the factors are then combined and multiplied by the learning rate for that iteration and applied to the signed difference of each of the attributes separately (line 16).

The training algorithm holistically pulls each of the cluster centers towards its neighboring high-density data points, where density is defined considering both distances and similarities between the data points and the cluster centers. This phase decreases the average distances between cluster centers and their respective members.

To determine the values of N and L, trial-and-error methods are recommended. The objective of the trials is to lower the mean intra-cluster average Euclidean distance, promoting tighter clusters. The cosine similarity should not be considered in this case, since the cluster centers will move towards the center of the neighboring high-density data points, which will average out the cosine similarity.

At the end of this phase, if multiple cluster centers become closer to the same dense data point neighborhood (i.e., Fig 3), they will overlap or become closer to each other. If a group of cluster centers has a marginal magnitude of separation at this stage, the Post-Processing phase, defined in Section III-D, merges them and produces a new cluster with a single cluster center. The following sections describe how to calculate the membership of a data point to a cluster, an important element of the Post-Processing phase.

C. MAPPING

The mapping algorithm labels a data point as a member of a cluster based on the cluster center nearest to it. The nearest cluster center is identified by the index corresponding to the minimum value of Δ_{ij}, which is the distance between data point i and cluster center j, calculated using the weighted Power Mean formula. We use both the Euclidean distance and cosine similarity in this Power Mean calculation, where each is normalized using maximum inter-cluster center distances and similarities, respectively. As described in Section III-A, V_N is used here since both measures are from two different metric spaces with different distances: Euclidean and cosine. The exact mathematical formula is shown in (4), where the value of Δ_{ij} is calculated for a cluster centre $\mathbf{C_i}$ and a data point $\mathbf{D_j}$ as follows:

$$\begin{aligned} e_{ij} &= \frac{||\mathbf{C_i} - \mathbf{D_j}||}{\max_{m,n}(||\mathbf{C_m} - \mathbf{C_n}||)} \\ c_{ij} &= \frac{\frac{\mathbf{C_i}\cdot\mathbf{D_j}}{||\mathbf{C_i}||||\mathbf{D_j}||}}{\max_{m,n} \frac{\mathbf{C_m}\cdot\mathbf{D_n}}{||\mathbf{C_m}||||\mathbf{D_n}||}} \\ a &= \frac{1}{1+V_N} \\ \Delta_{ij} &= \left((a\cdot e_{ij})^{\frac{1}{V_N}} + \left((1-a)\cdot c_{ij}\right)^{\frac{1}{V_N}}\right)^{V_N} \end{aligned} \tag{4}$$

The advantage of using the weighted Power Mean with V_N is it protects the Δ_{ij} from being over-biased by one domain. For example, when V_N is larger, the weighted Power Mean in the equation for Δ_{ij} can be shown to approach the geometric mean between the Euclidean distance and the cosine similarity. On the other hand, when V_N is smaller, the cosine similarity can be shown to be weighted more strongly. This makes sense intuitively, as the mean Euclidean distance will be larger in this case, meaning that small variations in cosine similarity will correspond to larger Euclidean distances between data points, and thus the cosine similarities should be weighted more strongly to reflect this fact.

D. POST-PROCESSING

Post-Processing is the final iterative process that merges the closest pair of clusters at each iteration. To determine the distance between a pair of clusters (e.g., Cl_i and Cl_j), the inter-cluster distance, $E(\mathbf{D_i}, \mathbf{C_j})$, is calculated, where $\mathbf{D_i}$ is a data point from cluster Cl_i and $\mathbf{C_j}$ is the cluster center for cluster Cl_j. This distance is calculated from the Euclidean distance, the cosine similarity, and the number of elements in the cluster pair. The minimum value $E(\mathbf{D_i}, \mathbf{C_j})$ found between any two clusters i and j is selected in each iteration and the two corresponding clusters are merged together.

The three factors that constitute the value for $E(\mathbf{D_i}, \mathbf{C_j})$ are: $E_d(\mathbf{D_i}, \mathbf{C_j})$, $E_s(\mathbf{D_i}, \mathbf{C_j})$, and $E_n(\mathbf{D_i}, \mathbf{C_j})$. The normalized distance between the cluster Cl_j having cluster center at $\mathbf{C_j}$ and the data point $\mathbf{D_i}$ belonging to cluster Cl_i, is defined to be $E_d(\mathbf{D_i}, \mathbf{C_j})$, while $E_s(\mathbf{D_i}, \mathbf{C_j})$ is the normalized similarity between the data point $\mathbf{D_i}$ and the cluster center $\mathbf{C_j}$. $E_n(\mathbf{D_i}, \mathbf{C_j})$ is equal to the sum of the square root of the number of elements in the cluster Cl_i and the square root of the number of elements in the cluster Cl_j. This latter factor is related to the distance between the two cluster centers $\mathbf{C_i}$ and $\mathbf{C_j}$, assuming both clusters have a circular shape in a normalized data set. In this case the radius of each of the cluster grows with the square root of the number of elements in the cluster, assuming the cluster elements are evenly spaced. Normalization is performed by dividing each value by respective maximum value over all ordered pairs of data points $\mathbf{D_i}$ and cluster centers $\mathbf{C_j}$. Theoretically, we expect the cluster radius to be roughly proportional to the nth root of the number of elements in the cluster for evenly-spaced n-dimensional data in a cluster. However, in practice when the dimension of the data is large, the nth root of the number of cluster elements will be close to unity regardless of the number of elements in the cluster. Therefore, we instead used the square root of the number of elements in the cluster, which worked well on the data sets we tested in this paper. The exact mathematical formula for calculating $E(\mathbf{D_i}, \mathbf{C_j})$ is given at (5).

$$\begin{aligned} E(\mathbf{D_i}, \mathbf{C_j}) &= F_d F_s F_n, \text{ where} \\ F_d &= 1 + \frac{E_d(\mathbf{D_i}, \mathbf{C_j})}{V_N} \\ F_s &= 1 + E_s(\mathbf{D_i}, \mathbf{C_j}) \\ F_n &= 1 + E_n(\mathbf{D_i}, \mathbf{C_j}) \end{aligned} \tag{5}$$

The merging of a pair of clusters in each iteration is followed by an update of the newly-formed cluster's center. In the next iteration, the new values of E_d, E_s and E_n are recalculated with the new set of cluster centers. At each iteration, the ratio between the maximum and minimum Euclidean distances among all the data points within a cluster is also calculated. The iterations continue until the step where the average ratio attains its maximum value. Therefore, the iterations must continue until one cluster remains in order to backtrack and correctly identify the right stopping iteration. An example of finding the correct stopping iteration is shown in Section IV-A where the algorithm is run on the *Iris flower* data set [45].

E. ALGORITHM COMPLEXITY

In this section, we determine the computational and space complexity of the proposed **Piecemeal Clustering** algorithm.

The Pre-Clustering phase initially calculates distances between all pairs of input data. The complexity of this operation is $O(n^2)$ and it generates an $n \times n$ symmetric matrix. To make the iterative process efficient, a *min-heap* can be created containing these n^2 values, which has a complexity of $O(n^2 \log n)$. Then the algorithm iterates at most n times, where in each iteration it searches for a minimum value on the heap. The searching has constant-time complexity since the minimum will always be at the top of the heap. In each iteration, a new data point also replaces two of the existing data points, therefore n new distance values are calculated while eliminating $2n$ data points from the heap. The worst case complexity for a sequence of n insertion and deletion operations can be calculated as $O(n \log n)$. Therefore, the overall complexity of this phase is $O(n^2 \log n)$.

The Training phase takes N outer iterations, where N is a user-defined value. In each iteration, it updates K ($K \leq n$) number of cluster centers at most n times. The value of N is much less than, and is independent of the number of data points for real-world data sets. Thus, it cannot be replaced with n. Therefore the worse case complexity for the Training phase is $O(Nn^2)$.

The Post-Processing phase also requires K iterations, where in each iteration it searches over n^2 values. Using a similar min-heap technique as mentioned above, we can complete this phase with a time complexity of $O(n^2 \log n)$.

Combining the complexity of all three phases, the overall time complexity for the algorithm is $O(n^2 \log n + Nn^2 + n^2 \log n)$. The value of N outweighs the the value of $\log n$. Therefore we determine the time complexity for the entire algorithm to be $O(Nn^2)$.

We determine the space complexity for this algorithm to be $O(n^2)$, since it generates n^2 all-pairs distance values that need to remain for all three phases.

IV. EXPERIMENTAL RESULTS

The algorithm was tested on two real world data sets: 1) *Iris flower*, a small data set with three known clusters, where two of the them are not linearly separable [46]; and 2) *Character Trajectories* [47], a large database of pen-tip trajectories captured while writing individual English characters. Applying the algorithm to the *Iris flower* data set demonstrates the correctness of the algorithm, whereas clustering the *Character Trajectories* data set shows the robustness of the algorithm to higher-dimensional data and outliers.

For both data sets, the value of T was varied incrementally from 0.01 to 1 with a step size of 0.01 until the final results did not greatly vary between one value of T and the next. The trials that used the maximum value of T without the result varying greatly are reported in the paper for both data sets. The results from each data set are given below.

TABLE 2. Numerical summary of *Iris Flower* data set.

Description	Value
Total number of data	150
Data Dimension	4
Dimension 1: Range of values	4.3 – 7.9
Dimension 2: Range of values	2.0 – 4.4
Dimension 3: Range of values	1.0 – 6.9
Dimension 4: Range of values	0.1 – 2.5

TABLE 3. Parameters used for clustering *Iris Flower* data set in Piecemeal Clustering algorithm.

Parameters	Value
T	0.15
N	120
L	0.0001

A. IRIS FLOWER DATA SET

The *Iris flower* data set consists of 50 samples from each of three species of Iris plants, present in the data in sequential order. Four features were measured from each of the samples. Table 2 shows a numerical summary of the data.

The data were prepared for the algorithm by normalizing each attribute to the interval [0,1] before applying the algorithm. This reduces the effect of different attribute variances on the clustering process. Normalization also normalizes the shape of the clusters in all dimensions. Table 3 shows the parameters used to cluster the data set.

The results after the Pre-Clustering and Training phases, as well as the final clustering, are shown in Fig 4. In each of the subfigures, data points are numbered sequentially starting from 1 and plotted on the horizontal axis. The cluster IDs are numbered sequentially and plotted on the vertical axis. The Pre-Clustering phase generates 17 clusters as shown in Fig 4a. However, we can also see that the 17 clusters are segmented into three groups. Next, the Training phase moves the cluster centers towards the most

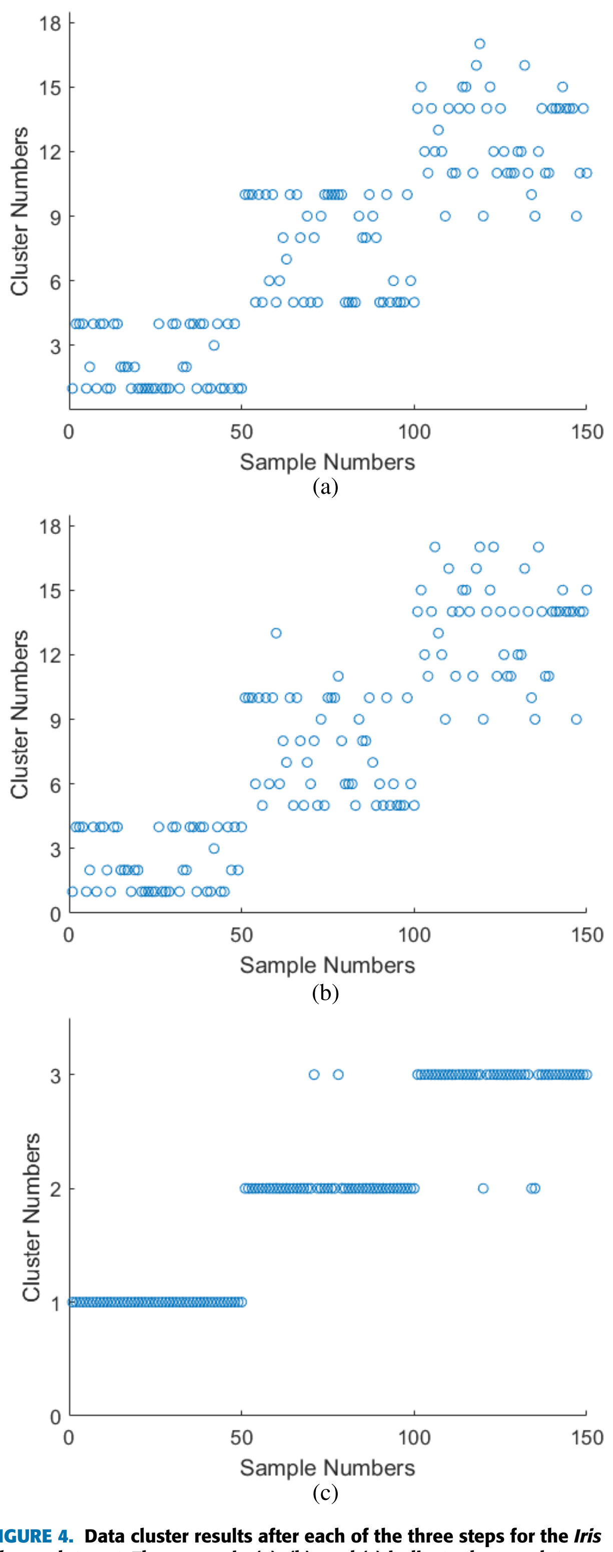


FIGURE 4. Data cluster results after each of the three steps for the *Iris Flower* data set. The x-axes in (a), (b), and (c) indicate the 150 data points, while the y-axes indicate the cluster IDs. The cluster IDs are natural numbers starting from 1 and are assigned arbitrarily to each of the observed clusters.

dense data points in their respective neighbourhoods, without considering membership of the points to the clusters. The result is shown in Fig 4b. Because of the change in the locations of the cluster centers, the data points now have different cluster memberships than in Fig 4a. This Training phase also prepares the data for Post-Processing. Finally, the Post-Processing phase merges one pair of clusters in each iteration, based on the algorithm described in Section III-D. Therefore, the algorithm continues for 16 iterations, leaving one cluster at the final iteration. At the end of each iteration, the new cluster centers are saved and the data points are mapped to the new cluster centers based on the method described in Section III-C. Next, for each cluster, the ratio of the maximum distance between any two elements of the cluster to the mean distance between any two elements of the cluster is determined. Logically, when two clusters are merged which should not have been, the mean distance between any two elements of the cluster will be higher than it should be since there will be a large number of samples from each cluster which is far from many other samples in the other cluster just merged. Thus if we take the mean of this ratio over all clusters in the new clustering, then it should steadily increase up to the point that a good clustering is found, and then decrease immediately after that. Therefore, we took the cluster centers corresponding to this maximum ratio as the ones corresponding to the final clustering.

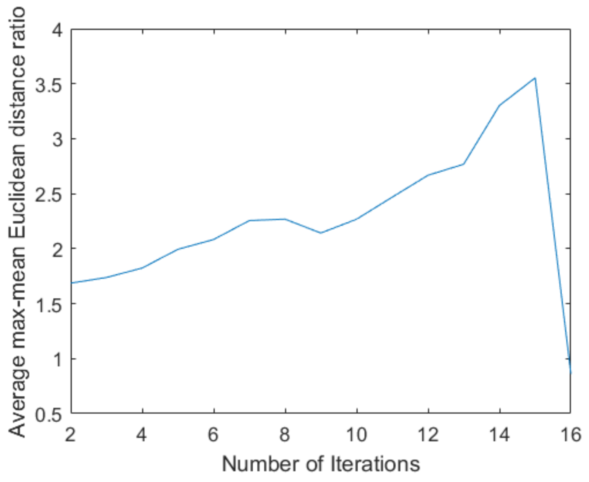


FIGURE 5. Change in average of pairwise maximum-to-mean Euclidean distance ratio over Post-Processing iterations.

At the end of 16 iterations, the ratio values for each iterations are plotted in Fig 5. The horizontal axis represents the iteration number, while the vertical axis represents the mean ratio. Since the ratio peaked at iteration 15, the saved cluster centers from the end of iteration 14 are chosen, as they were used to calculate the ratio in iteration 15, and the data points are mapped to these cluster centers. This generates three clusters, as shown in Fig 4c. The final result shows 145 data points out of 150 are clustered properly, giving an overall accuracy of 100% for finding the correct total number of clusters in the data and an accuracy of 96.7% for mapping data points to their correct clusters for the **Piecemeal Clustering** algorithm.

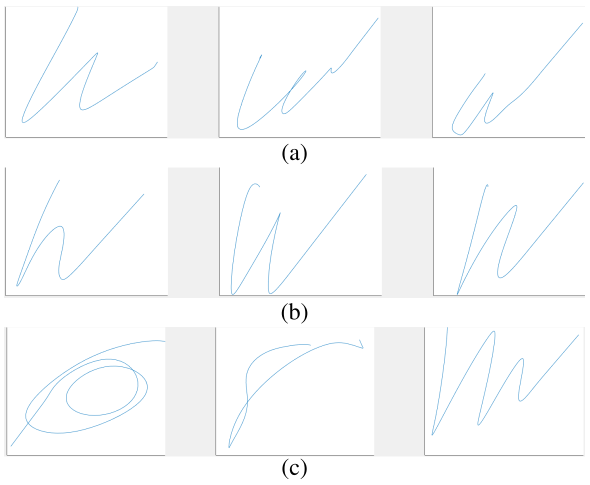


FIGURE 6. Example of diverse visual appearances of the instances of the characters in the *Character Trajectory* data set. (a) Three instances of the English letter w where all of them differ in appearance, (b) three instances of the English letter h, u, and n, which all look similar, and (c) three character instances that are hard to identify as any English letter.

B. CHARACTER TRAJECTORIES DATA SET

The *Character Trajectories* data set consists of 2858 samples of pen tip trajectories recorded whilst writing individual English lower case letter characters. The original data collection consisted of a series of two-dimensional coordinates (x, y), and pen tip force on a WACOM tablet, sampled at a rate of 200 Hz. Only letters with a single pen-down segment were considered for that experiment. Therefore a total of 20 letters were included in the data sets, and the letters: 'f', 'i', 'j', 'k', 't', 'x' were excluded. The objective of our experiment was to cluster the data sets based on their trajectories. Therefore, only the Cartesian coordinates were used, and the pen tip force data were excluded from consideration.

The data set consists of a variety of visually diverse character instances. Some character instances were written for a single letter, but were very different in appearance. Some other instances appear to be similar, but were written for different letters. And there were instances that were hard to recognize as any English letter. Fig 6 shows examples of the diverse visual appearance of character instances in the data set. Fig 6a shows three instances of the letter w. Fig 6b shows three instances of the letters h, u, and n from left to right, which all look very similar to the letter w. Fig 6c shows three character instances that are very difficult to identify. The diverse visual appearance of the instances makes the proposed **Piecemeal Clustering** algorithm a good candidate to determine a natural clustering of the data, where each cluster will consist of character instances that are visually similar.

The character instances in the data set consisted of a maximum of 205 points in the Cartesian plane. Therefore, each instance had a total of at most 410 attributes. As a result, the robustness of the algorithm can be tested using this data set, both in terms of the number of attributes and number of data points in the data set.

TABLE 4. Clustering parameters for *Character Trajectories* data set.

Parameters	Value
T	0.14
N	65
L	0.00005

A pre-processing step was performed on the data before applying the algorithm. As previously mentioned, the data set consisting of the trajectory of the pen tip was collected with a constant sampling frequency. Therefore, the trajectory lengths for each character instance were not necessarily the same. The characters were also written in different quadrants with different scaling. Hence, the trajectory length of each of the symbols was normalized using linear interpolation. We also found that not all the characters were drawn at the same scale, which is natural for handwriting. To bring all the character instances into the same scale and quadrant, they were normalized to values between 0 and 1 along both the x and y axes. Principal Component Analysis (PCA) was then applied to the data to increase the stability of the clusters [48].

After pre-processing of the data, the algorithm was applied with the parameters shown in Table 4. The Pre-Clustering phase generated 188 clusters. Each of these clusters was smaller in size with their samples visually similar to each other. After passing through the Training and Post-Processing phases, a total of 20 clusters were generated.

We compared the mapping of individual data points using a confusion matrix as shown in the Fig 7 with the known result of the data set. We labeled the 20 clusters based on the dominating characters instances. For example, we found a cluster containing 219 character instances (data points); among them, 167 were identified as 'a', 17 as 'd', 1 as 'n', 31 as 'o', and 3 as 'u' in the known result. We labeled this cluster as 'a' since this character is found most in this cluster. In Fig 7, this cluster is in column 1. We repeated the process for all 20 predicted clusters. We were able to find one cluster for each of the 20 letters in the data set and name them accordingly. We placed them in the confusion matrix in the order from left to right, in the same sequence as the dominating letters appear in the English alphabet. Since the **Piecemeal Clustering** was able to find the exact 20 clusters as found in the data set, the proposed algorithm is 100% accurate in finding the number of clusters. The varying colour density in the diagonal of the confusion matrix also demonstrates that the proposed algorithm can find clusters of varying sizes.

As discussed above and illustrated in Fig 6, it is difficult to avoid mapping character instances to a cluster where the dominating character is not a match. We counted such mismatches in each cluster. We found that 13 clusters had zero to ten instances of mismatches. The rest of the clusters had a 9-38% mismatch in their mappings. The seven clusters with highest mismatches present were: 'a', 'c', 'g', 'l', 'n', 'u', 'w', with the details shown in Table 5. Some of these

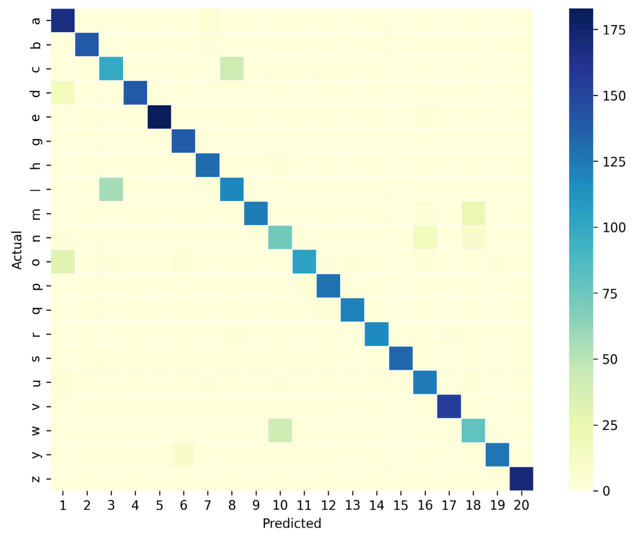


FIGURE 7. **Confusion matrix comparing the mapping of *Character Trajectories* data set with true result. The diagonal represents the number of data points accurately mapped by the Piecemeal Clustering algorithm. The number of elements in each cell is coded with varying colour density. The darker the cell, the higher the value it represents.**

TABLE 5. **Clusters with highest mismatch after clustering the *Character Trajectories* data set.**

Description	Count
'c' misclassified as 'l'	56
'l' misclassified as 'c'	42
'n' misclassified as 'w'	41
'a' misclassified as 'o'	31
'w' misclassified as 'm'	24
'u' misclassified as 'n'	17
'a' misclassified as 'd'	17
'g' misclassified as 'y'	10
'w' misclassified as 'n'	10

mixes are expected for the natural handwriting curvature of hand written small letter English characters, such as: 'a' with 'd' or 'o', 'c' with 'l', 'g' with 'y', 'm' with 'w', and 'n' with 'u'. We also noticed an unexpected mix, 'n' with 'w'. The cluster labeled as 'z' stands out among all of these clusters. This cluster separated all the character instances of 'z' without any mix with any other clusters, and no other character instances were mixed with 'z' either. 's' is another such cluster. We counted all the incorrectly mapped character instances to calculate the mapping efficiency of **Piecemeal Clustering**. A total of 281 instances was mismatched, leaving 2,577 mapped correctly. Therefore, our algorithm shows an accuracy of 90.2% in clustering on this difficult high-dimensional data set.

V. COMPARISONS

We compared the performance of the proposed **Piecemeal Clustering** algorithm with *K-means*, *SOM*, *Hierarchical*, *DBSCAN*, *RNN-DBSCAN*, *HDBSCAN**, and *Blocked-DBSCAN*. The first four clustering algorithms are generic and widely used in various fields of study. The other three are more advanced and have been proposed within the last decade to overcome some of the limitations of the generic algorithms. Clustering was performed with these seven algorithms on both the *Iris Flower* and *Character Trajectory* data sets. The accuracy of finding the correct number of clusters, the degree of accuracy of mapping data to each of their respective clusters, and the total numbers of correctly mapped data points are compared. Subjective analyses were also performed to understand the incorrect mapping.

The MATLAB® implementations of *K-means*, *SOM*, *Hierarchical*, and *DBSCAN* algorithms from Release 2019a with the Machine Learning toolbox were used to generate results. The MATLAB® functions that were used were *kmeans*, *selforgmap*, *clusterdata*, and *dbscan* for their respective algorithms. For *RNN-DBSCAN* we used the RNN DBSCAN package [49] from the MATLAB® Library. The Scikit Learn Python package [50] was used for HDBSCAN*. Author-provided C++ code was used for *Blocked-DBSCAN*.

K-means and *SOM* both require prior knowledge of the number of clusters; therefore, such information was provided when clustering both of the data sets. The *Hierarchical* algorithm can cluster data with and without prior knowledge of the total number of clusters. Both methods were tried. *DBSCAN* and *Blocked-DBSCAN* can work without any prior knowledge of the data. We used a *k-distance* graph to determine the values of the ϵ parameter of *DBSCAN* and used prior knowledge of the data sets to limit the max value for the *minimum number of elements* in each cluster. For RNN-DBSCAN we applied an exhaustive search method using *KNN-Search* for both data sets. *HDBSCAN** was configured to use an approximation of the minimum spanning tree with varying α and *leafsize* to cluster the data sets.

The parameters of each algorithm were optimized to obtain the best results. The set of parameters giving the maximum number of correctly clustered data points was considered as the best result for an algorithm. The results are compared for the data sets in the the following sections.

A. IRIS FLOWER DATA SET

K-means clustering was performed with the parameters K=3 and *Replicates* = 1000 in the MATLAB® *kmeans* function. A variety of distance metrics was tested with different numbers of iterations. The squared Euclidean distance yielded the best result with 100 iterations.

For *SOM*, 1-by-3, 3-by-1, and 2-by-2 maps were tested with different network topologies. A wide range of numbers of iterations was also tested with both Euclidean distance and cosine similarity as distance metrics. Initial neighborhood sizes of 1, 2 and 3 were tested in combination with the above settings. The best result was found with a 1-by-3 network, *hextop* network topology, Euclidean distance for the distance calculation, initial neighborhood size 2, and 50 iterations.

For the *Hierarchical* algorithm, a brute force approach was used for *cutoff* from 0 to 2. A *cutoff* value of 3 was also used to identify a maximum of 3 clusters. All these cutoff values were tried with both Euclidean and cosine similarity as distance metrics, and different combinations of *linkage* methods. The best result found was with the Euclidean distance metric, single linkage, and a cutoff value of 3.

With *DBSCAN* and *Block-DBSCAN*, brute force approaches were applied to find the best selection for the values of ϵ and *numpts*. The brute force method searched for all the values between 0.1 and 2.0 with step size of 0.00001 for ϵ, and all the integers between 1 and 40 for *numpts*. The maximum allowed value 2.0 for ϵ was determined by using the k-distance graph. The values $\epsilon = 0.159$, $minpts = 16$, and Euclidean distance yielded the best result for *DBSCAN*, while $\epsilon = 0.242$, $minpts = 10$, and Euclidean distance yielded the best result for *Blocked-DBSCAN*.

A brute force approach was also used for *RNN-DBSCAN* with the maximum value of 20 for *nNeighborsIndex*. *nNeighbors* = 6 yielded the best results. For *HDBSCAN** an α value between 0.1 and 1.0 with step size of 0.001 was searched for. The values $\alpha = 0.5$, *leafsize* = 2, and *minkowski* for pairwise distance yielded the best result.

Table 6 shows a comparison of the algorithms. Except for the first row, all the rows in the table represent a result from an algorithm mentioned in the first column. The following five columns are used for objective comparison of the results, while the last column is used for subjective analysis.

In Table 6, the "Number of Identified Clusters" and "Number of Unique Clusters" columns are used to understand the accuracy of **Piecemeal Clustering** to find the correct number of clusters. All the algorithms, except *K-means*, can produce any number of clusters. We observe that *Hierarchical* and *HDBSCAN** identified only two clusters from the data. *Block-DBSCAN* produced four clusters. We labeled all the identified clusters based on the number of dominating members, described in Section IV-B. We noticed that two algorithms, *RNN-DBSCAN* and *Blocked-DBSCAN*, produced more than one cluster from one known cluster. For instance, *RNN-DBSCAN* produced two clusters from the first 50 data points, where all these data points are part of the same cluster in the known result. The third identified cluster contained the last 100 data points, wherein the one hundred data points were part of two clusters in the known result. *RNN-DBSCAN* also identified seven data points from the first 50 as noise, mentioned in the "Size of Noise" column in the table. When any algorithm produced more than one cluster from the same clusters in the known result, we merged them and considered them as one unique cluster for the purpose of this analysis.

We know the last 100 data points were mapped to two clusters but are not linearly separable. *RNN-DBSCAN* and *HDBSCAN** were not able to separate them. On the other hand, the other algorithms, including **Piecemeal Clustering**, were able to separate them successfully. *Blocked-DBSCAN* was able to identify three unique clusters while splitting the entire data into four clusters. On the other hand, *DBSCAN* detected no noise in the data set, which is aligned with the known result. *RNN-DBSCAN*, *HDBSCAN** and *Blocked-DBSCAN* incorrectly identified some data points as noise.

Piecemeal Clustering showed similar performance in finding the correct number of clusters and the correct unique number of clusters to the other three algorithms: *K-means*, *SOM*, and *DBSCAN*. In contrast, **Piecemeal Clustering** outperformed all seven algorithms in mapping the data points correctly, mapping 145 out of 150 data points correctly (96.7% accuracy). The incorrect mapping is consistent across all the algorithms. All algorithms could uniquely identify the first 50 data points and mismatched other points in the other two clusters.

B. CHARACTER TRAJECTORY DATA SET

K-means clustering was performed with the parameters K=20 and *Replicates* = 1000. A variety of distance metrics was tested with different numbers of iterations. The squared Euclidean distance yielded the best result with 100 iterations.

Like *K-means*, different sets of parameters were selected for *SOM* to cluster the data set. Both 4-by-5 and 5-by-4 networks were tested along with different network topologies, distance metrics, and numbers of iterations. The best result was found with a 4-by-5 network, *hextop* network topology, Euclidean distance, initial neighborhood size of 3, and 100 iterations.

For the *Hierarchical* algorithm, finely-quantized *cutoff* values between 0 to 2 were tested with both the Euclidean distance and cosine similarity as distance metrics. The minimum number of clusters produced using the Euclidean distance was 793, while with cosine similarity, 472 clusters were produced. Since both of these approaches produced a large number of clusters, their results were incomparable with the other results. Next, a *cutoff* value of 20 was tested with both the Euclidean distance and cosine similarity. Both of the distance metrics yielded nine significant clusters. Since both the Euclidean distance and cosine similarity metrics provided similar results, the result generated using the Euclidean distance metric was used for comparison.

A brute force approach was used to search for ϵ between 0 to 5 with step size 0.00001, and all integer values between 0 and 100 were tested for the value of *numpts*, for *DBSCAN* and *Block-DBSCAN* clustering. Different distance metrics were used in combination with the ϵ and *numpts* values. For *DBSCAN*, the best result was found for an ϵ value of 0.0061, *numpts* value of 6 and cosine similarity. For *Blocked-DBSCAN*, a combination of ϵ value of 2.3, *numpts* equal to 6, and Euclidean distance produced the best result.

A brute force approach was also used for *RNN-DBSCAN* with a maximum value of 20 for *nNeighborsIndex*, and *nNeighbors* = 7 yielding the best results. For *HDBSCAN**, α values between 0.1 and 1.0 with step size of 0.001 were searched. $\alpha = 0.9$, *leafsize* = 5, and *minkowski* for pairwise distance yielded the best result.

TABLE 6. Performance comparison of clustering algorithms on *Iris Flower* data set. Highest accuracies and correct number of noise samples are highlighted.

Algorithms	Number of Identified Clusters	Number of Unique Clusters	Number of Correctly Mapped Data Points	Mapping Accuracy	Size of Noise	Incorrect Mapping[a]
Piecemeal	3	**3**	**145**	**96.7%**	-	(2,3)
K-means	3	**3**	133	88.7%	-	(2,3)
SOM	3	**3**	133	88.7%	-	(2,3)
Hierarchical	2	2	100	66.7%	-	(2,3)
DBSCAN	3	**3**	128	85.3%	**0**	(2,3)
RNN-DBSCAN	3	2	93	62.0%	7	(2,3)
HDBSCAN*	2	2	98	65.3%	4	(2,3)
Blocked DBSCAN	4	**3**	116	77.3%	2	(2,3)

[a]The clusters are labeled as 1, 2, and 3 where cluster 1 is linearly separable and 2 and 3 are not. This column shows how the incorrect mapping spreads over the clusters

TABLE 7. Comparison of performance of clustering algorithms on *Character Trajectory* data set. Highest accuracies are highlighted.

Algorithms	Number of Identified Clusters	Number of Unique Clusters	Number of Correctly Mapped Data Points	Mapping Accuracy	Size of Noise	Incorrect Mapping[a]
Piecemeal	20	**20**	**2577**	**90.2%**	-	('a','o'),('c','l'),('n','u'),('n','w'), ('m','w') ('a','d'), ('g','y')
K-means	20	18	2386	83.5%	-	('g','y'), ('u','n')
SOM	20	18	2366	82.8%	-	('g','y'),('u','w')
Hierarchical	12	8	1228	43.0%	-	N/A[b]
DBSCAN	20	17	1771	62.0%	769	('g','y') ('h','c')
RNN-DBSCAN	20	18	2295	80.3%	156	('h','n') , ('r','v')
HDBSCAN*	20	18	2402	84.0%	245	('h','n'),('b','p')
Blocked DBSCAN	20	14	1812	63.4%	5	('c','l'), ('e','g','y'), ('h','r'), ('a','h','m','n','o','u','w')

[a]The clusters are labeled as their representative letter in the alphabet.
[b]This is not calculated because of the lower number of unique clusters.

Results from all tested algorithms are summarized in Table 7. As with the *Iris Flower* data set experiment, all the algorithms were set to produce an arbitrary number of clusters, except *K-means*. All but the *Hierarchical* algorithm produced 20 clusters. The clusters were labeled with the letters in the alphabet present the most in the data set. Only **Piecemeal Clustering** produced 20 unique clusters, one for each of the letters. **Piecemeal Clustering** also outperformed the other algorithms in mapping the data points correctly with an overall accuracy of 90.2%. *K-means*, *SOM*, *RNN-DBSCAN* and *HDBSCAN** also produced good results, except they each, in their individual results, incorrectly combined pairs of distinct clusters into one.

Further exploration of the results of these four algorithms reveals merging of clusters could be expected for *K-means*, *SOM*, *RNN-DBSCAN* because of the curvature of the participating character instances. For *HDBSCAN**, one such merging is expected, ('h','n'); while the other one was not expected, ('b','p'). Nonetheless, in terms of mapping *HDBSCAN** produced the second-best result after **Piecemeal Clustering**.

While **Piecemeal Clustering** could produce the correct number of clusters, separating the letters as unique clusters, and mapping the highest number of data points, incorrect mappings were found in more clusters than from some of the other algorithms. In this comparison, the *K-means*, *SOM*, *DBSCAN*, *RNN-DBSCAN*, and *HDBSCAN** algorithms were more successful in isolating more clusters without mixing with other letters. At the same time, we also observed that the small number of incorrect mappings (281 and 456 incorrect mapping by **Piecemeal Clustering** and *HDBSCAN**, respectively) in the **Piecemeal Clustering** can be expected because of the similar curvatures shared by character instances across the clusters.

C. STATISTICAL SIGNIFICANCE

To understand the statistical significance of the results generated by the **Piecemeal Clustering** algorithm over the other algorithms, we ran the McNemar Tests [51]. Three cost-insensitive tests were performed to compare the results from our study with the results found from the algorithms used in the comparison. The three tests are: asymptotic

TABLE 8. Statistical significance of Piecemeal Clustering result on *Iris Flower* data set vs. the other algorithms.

Algorithm	h	Asymptotic	Exact	Mid-p
K-means	1	6.70×10^{-4}	4.88×10^{-4}	9.15×10^{-4}
SOM	1	6.70×10^{-4}	4.88×10^{-4}	9.15×10^{-4}
Hierarchical	1	1.48×10^{-10}	5.21×10^{-12}	9.84×10^{-12}
DBSCAN	1	1.97×10^{-4}	1.39×10^{-4}	2.44×10^{-4}
HDBSCAN*	1	2.33×10^{-11}	3.06×10^{-13}	5.89×10^{-13}
RNN-DBSCAN	1	4.30×10^{-12}	5.95×10^{-14}	1.13×10^{-13}
Blocked DBSCAN	1	4.74×10^{-7}	1.14×10^{-7}	2.09×10^{-7}

TABLE 9. Statistical significance of Piecemeal Clustering result on *Character Trajectory* data set vs. the other algorithms.

Algorithm	h	Asymptotic	Exact	Mid-p
K-means	1	2.48×10^{-22}	2.80×10^{-23}	4.27×10^{-22}
SOM	1	7.88×10^{-25}	3.72×10^{-26}	5.85×10^{-26}
DBSCAN	1	1.27×10^{-147}	1.48×10^{-173}	2.74×10^{-173}
HDBSCAN*	1	1.22×10^{-11}	9.36×10^{-12}	1.21×10^{-11}
RNN-DBSCAN	1	6.77×10^{-17}	3.34×10^{-17}	4.55×10^{-17}
Blocked DBSCAN	1	1.25×10^{-126}	5.98×10^{-143}	1.05×10^{-142}

test, the exact-conditional test, and the mid-p-value test. We calculated the h and p values for all the tests. The h value was calculated at the 0.05 significance level. The p for the tests for the *Iris Flower* and *Character Trajectory* data sets are shown in Table 8 and Table 9, respectively. We compare the results for Iris data set for all seven algorithms, while we skip results from *Hierarchical* for *Character Trajectory* data set, since it was not generating comparable results as shown in Table 7. According to the McNemar tests, a lower value of p indicates more statistical significance.

The null hypothesis is rejected for all the tests in both data sets, which indicates that the **Piecemeal Algorithm** has better accuracy than the other algorithms compared. All the p values are close to zero for all three tests, which further emphasizes the null value rejections.

We also noticed from Table 6, for the *Iris Flower* data set, *K-means* provides the second best results after **Piecemeal Clustering** in terms of mapping the data points to their respective clusters. At the same time, the statistical significance of the results from *K-means* algorithm is comparatively less than the *DBSCAN* algorithm, an alignment with the literature as found in Table 1. For the *Character Trajectory* data set, *HDBSCAN** is consistent in both accuracy count and statistical significance comparison.

VI. CONCLUSION

A novel and robust data clustering algorithm, **Piecemeal Clustering**, is described in this paper. It is an unsupervised-learning based algorithm combining ideas from *Hierarchical* clustering, *SOM*, and *DBSCAN*. It is a generic algorithm that can robustly handle variability within clusters. In particular, it can correctly identify the number of clusters from the natural clustering of both higher- and lower-dimensional data, even if the clusters are not linearly separable. In this paper, the algorithm was tested on two data sets, with both small and large numbers of attributes and uneven cluster sizes. It was also benchmarked against established and recently proposed algorithms. The **Piecemeal Clustering** algorithm outperforms all the algorithms on both the *Iris Flower* and *Character Trajectory* data sets, identifying the correct number of clusters and mapping data points to the correct clusters. The results were also found to be statistically significant when compared with the other tested algorithms.

While the proposed algorithm provides good clustering results without prior knowledge of the number of clusters in the data, it has some limitations. Normalizing the attributes to provide equal weight to all of the attributes is key to the result. Therefore pre-processing of the data is required before applying the algorithm. The algorithm has slightly higher time complexity than some of the algorithms in literature. The algorithm also uses three parameters. Like any machine learning algorithm, determining the learning rate and number of training iterations in the learning phase of the algorithm is a trial-and-error procedure, and any trial cannot be validated without executing the Post-Processing phase. Both Pre-Clustering and Post-Processing phases are iterative, and therefore, may take a long time to compute for large data sets.

In the future, the proposed algorithm can be improved by automating the selection of some or all of its parameters. Mathematical equations can be explored to determine the optimum cutoff threshold, T. Further studies can also be conducted on the impact of batch processing on the Pre- and Post-Processing phases on the quality of clusters obtained by the algorithm. Batch processing can reduce the time complexity of the algorithm significantly. More methods can also be explored for optimizing the time complexity of the algorithm. A parallel version of this proposed algorithm may also be developed for clustering big data. The **Piecemeal Clustering** algorithm may also be applied to different fields of study, e.g., identifying lithofacies from well logs in oil field exploration and generating customer clusters based on their shopping patterns.

ACKNOWLEDGMENT

The authors would like to thank Memorial University for logistical support.

REFERENCES

[1] A. C. Benabdellah, A. Benghabrit, and I. Bouhaddou, "A survey of clustering algorithms for an industrial context," *Proc. Comput. Sci.*, vol. 148, pp. 291–302, Jan. 2019.
[2] C. Pan and J. Tan, "Day-ahead hourly forecasting of solar generation based on cluster analysis and ensemble model," *IEEE Access*, vol. 7, pp. 112921–112930, 2019.

[3] C. Kolluru, J. Lee, Y. Gharaibeh, H. G. Bezerra, and D. L. Wilson, "Learning with fewer images via image clustering: Application to intravascular OCT image segmentation," *IEEE Access*, vol. 9, pp. 37273–37280, 2021.

[4] C.-E. B. Ncir, A. Hamza, and W. Bouaguel, "Parallel and scalable Dunn index for the validation of big data clusters," *Parallel Comput.*, vol. 102, May 2021, Art. no. 102751.

[5] W. Qi, Q. Song, X. Wang, L. Guo, and Z. Ning, "SDN-enabled social-aware clustering in 5G-VANET systems," *IEEE Access*, vol. 6, pp. 28213–28224, 2018.

[6] J. Majumdar, S. Udandakar, and B. M. Bai, "Implementation of cure clustering algorithm for video summarization and healthcare applications in big data," in *Emerging Research in Computing, Information, Communication and Applications*, N. R. Shetty, L. M. Patnaik, H. C. Nagaraj, P. N. Hamsavath, and N. Nalini, Eds. Singapore: Springer, 2019, pp. 553–564.

[7] D. Huang, C.-D. Wang, J.-S. Wu, J.-H. Lai, and C.-K. Kwoh, "Ultra-scalable spectral clustering and ensemble clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 6, pp. 1212–1226, Jun. 2020.

[8] I. Annaki, M. Rahmoune, M. Bourhaleb, N. Rahmoun, M. Zaoui, A. Castilla, A. Berthoz, and B. Cohen, "Computational analysis of human navigation in a VR spatial memory locomotor assessment using density-based clustering algorithm of applications with noise DBSCAN," in *Digital Technologies and Applications*, S. Motahhir and B. Bossoufi, Eds. Cham, Switzerland: Springer, May 2022, pp. 190–198.

[9] J.-S. Lee, H.-T. Lee, and I.-S. Cho, "Maritime traffic route detection framework based on statistical density analysis from AIS data using a clustering algorithm," *IEEE Access*, vol. 10, pp. 23355–23366, 2022.

[10] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Berkeley Symp. Math. Statist. Probab.* Berkeley, CA, USA: Univ. of California Press, 1967, pp. 281–297.

[11] E. Schubert and P. J. Rousseeuw, "Fast and eager k-medoids clustering: O(k) runtime improvement of the PAM, CLARA, and CLARANS algorithms," *Inf. Syst.*, vol. 101, Nov. 2021, Art. no. 101804.

[12] D. P. Ismi and M. Murinto, "Clustering based feature selection using partitioning around medoids (pam)," *Jurnal Informatika*, vol. 14, no. 2, pp. 50–57, May 2020.

[13] K. Indira, S. Karthiga, C. V. N. Angeline, and C. Santhiya, "Parallel CLARANS algorithm for recommendation system in multi-cloud environment," in *Computer Networks and Inventive Communication Technologies*, S. Smys, R. Palanisamy, Á. Rocha, and G. N. Beligiannis, Eds. Singapore: Springer, 2021, pp. 461–472.

[14] C. E. Rasmussen, "The infinite Gaussian mixture model," in *Proc. Int. Conf. Neural Inf. Process. Syst.* Cambridge, MA, USA: MIT Press, 1999, pp. 554–560.

[15] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.

[16] D. K. Jain, S. B. Dubey, R. K. Choubey, A. Sinhal, S. K. Arjaria, A. Jain, and H. Wang, "An approach for hyperspectral image classification by optimizing SVM using self organizing map," *J. Comput. Sci.*, vol. 25, pp. 252–259, Mar. 2018.

[17] H.-C. Yang, C.-H. Lee, and C.-Y. Wu, "Sentiment discovery of social messages using self-organizing maps," *Cogn. Comput.*, vol. 10, no. 6, pp. 1152–1166, Dec. 2018.

[18] A. Huang and F.-J. Chang, "Using a self-organizing map to explore local weather features for smart urban agriculture in northern Taiwan," *Water*, vol. 13, no. 23, p. 3457, Dec. 2021.

[19] X. Zheng, X. Yang, H. Miao, H. Liu, Y. Yu, Y. Wang, H. Zhang, and S. You, "A factor analysis and self-organizing map based evaluation approach for the renewable energy heating potentials at county level: A case study in China," *Renew. Sustain. Energy Rev.*, vol. 165, Sep. 2022, Art. no. 112597.

[20] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. Int. Conf. Knowl. Discovery Data Mining*. Palo Alto, CA, USA: AAAI Press, 1996, pp. 226–231.

[21] A. E. Ezugwu, A. M. Ikotun, O. O. Oyelade, L. Abualigah, J. O. Agushaka, C. I. Eke, and A. A. Akinyelu, "A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects," *Eng. Appl. Artif. Intell.*, vol. 110, Apr. 2022, Art. no. 104743.

[22] M. W. Dunham, A. Malcolm, and J. K. Welford, "Improved well log classification using semisupervised Gaussian mixture models and a new hyper-parameter selection strategy," *Comput. Geosci.*, vol. 140, Jul. 2020, Art. no. 104501.

[23] J. H. Ward, Jr., "Hierarchical grouping to optimize an objective function," *J. Amer. Statist. Assoc.*, vol. 58, no. 301, pp. 236–244, 1963.

[24] C. Bouveyron and C. Brunet-Saumard, "Model-based clustering of high-dimensional data: A review," *Comput. Statist. Data Anal.*, vol. 71, pp. 52–78, Mar. 2014.

[25] M. Roux, "A comparative study of divisive and agglomerative hierarchical clustering algorithms," *J. Classification*, vol. 35, no. 2, pp. 345–366, Jul. 2018.

[26] F. Alalyan, N. Zamzami, and N. Bouguila, "Model-based hierarchical clustering for categorical data," in *Proc. IEEE 28th Int. Symp. Ind. Electron. (ISIE)*, Jun. 2019, pp. 1424–1429.

[27] W.-B. Xie, Y.-L. Lee, C. Wang, D.-B. Chen, and T. Zhou, "Hierarchical clustering supported by reciprocal nearest neighbors," *Inf. Sci.*, vol. 527, pp. 279–292, Jul. 2020.

[28] Y. Ma, H. Lin, Y. Wang, H. Huang, and X. He, "A multi-stage hierarchical clustering algorithm based on centroid of tree and cut edge constraint," *Inf. Sci.*, vol. 557, pp. 194–219, May 2021.

[29] J. Shi, Q. Zhu, and J. Li, "A novel hierarchical clustering algorithm with merging strategy based on shared subordinates," *Int. J. Speech Technol.*, vol. 52, no. 8, pp. 8635–8650, Jun. 2022.

[30] R. Petegrosso, Z. Li, and R. Kuang, "Machine learning and statistical methods for clustering single-cell RNA-sequencing data," *Briefings Bioinf.*, vol. 21, no. 4, pp. 1209–1223, Jul. 2020.

[31] M. Alloghani, D. Al-Jumeily, J. Mustafina, A. Hussain, and A. J. Aljaaf, "A systematic review on supervised and unsupervised machine learning algorithms for data science," in *Supervised and Unsupervised Learning for Data Science*. Cham, Switzerland: Springer, Sep. 2020, pp. 3–21.

[32] J.-H. Kim, J.-H. Choi, K.-H. Yoo, and A. Nasridinov, "AA-DBSCAN: An approximate adaptive DBSCAN for finding clusters with varying densities," *J. Supercomput.*, vol. 75, no. 1, pp. 142–169, Jan. 2019.

[33] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," in *Proc. Int. Conf. Data Eng.* Washington, DC, USA: IEEE Computer Society, 1998, pp. 324–331.

[34] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial–temporal data," *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, 2007.

[35] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Advances in Knowledge Discovery and Data Mining*, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Berlin, Germany: Springer, 2013, pp. 160–172.

[36] A. A. Bushra and G. Yi, "Comparative analysis review of pioneering DBSCAN and successive density-based clustering algorithms," *IEEE Access*, vol. 9, pp. 87918–87935, 2021.

[37] A. Bryant and K. Cios, "RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 6, pp. 1109–1121, Jun. 2018.

[38] L. McInnes and J. Healy, "Accelerated hierarchical density based clustering," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Piscataway, NJ, USA, Nov. 2017, pp. 33–42.

[39] Y. Chen, L. Zhou, N. Bouguila, C. Wang, Y. Chen, and J. Du, "BLOCK-DBSCAN: Fast clustering for large scale data," *Pattern Recognit.*, vol. 109, Jan. 2021, Art. no. 107624.

[40] D. Xu and Y. A. Tian, "A comprehensive survey of clustering algorithms," *Ann. Data Sci.*, vol. 2, no. 2, pp. 165–193, Jun. 2015.

[41] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3336–3341, Mar. 2009.

[42] C. Malzer and M. Baum, "A hybrid approach to hierarchical density-based cluster selection," in *Proc. IEEE Int. Conf. Multisensor Fusion Integr. Intell. Syst. (MFI)*, Sep. 2020, pp. 223–228.

[43] G. Stewart and M. Al-Khassaweneh, "An implementation of the HDBSCAN* clustering algorithm," *Appl. Sci.*, vol. 12, no. 5, p. 2405, Feb. 2022.

[44] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.

[45] R. Fisher. (1936). *Iris Data Set*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/iris

[46] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, Sep. 1936.

[47] B. H. Williams. (2006). *Character Trajectories Data Set*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Character+Trajectories

[48] A. Ben-Hur and I. Guyon, "Detecting stable clusters using principal component analysis," in *Functional Genomics: Methods and Protocols*, M. J. Brownstein and A. B. Khodursky, Eds. Totowa, NJ, USA: Humana Press, 2003, pp. 159–182.

[49] T. Vannoy. (2022). *RNN DBSCAN*. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/97924-rnn-dbscan

[50] L. McInnes, J. Healy, and S. Astels, "Hdbscan: Hierarchical density based clustering," *J. Open Source Softw.*, vol. 2, no. 11, p. 205, Mar. 2017.

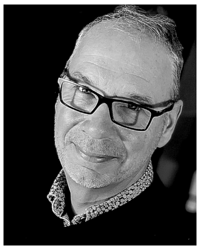
[51] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, Jun. 1947.

MD. MONJUR UL HASAN received the B.Sc. degree in computer science and engineering from the Chittagong University of Engineering and Technology, Bangladesh, in 2005, and the M.Sc. degree in geovisual analytics from the Memorial University of Newfoundland, Canada, in 2015, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. His research interests include data clustering, data analytics, and machine learning.

REZA SHAHIDI (Senior Member, IEEE) is currently an Adjunct Professor and a Teaching Assistant Professor with the Faculty of Engineering and Applied Science, Memorial University of Newfoundland. He is the author of over 40 peer-reviewed journals and conference publications. His research interests include algorithm design, software engineering, machine learning, signal processing, image processing, radar, remote sensing, and seismic imaging.

DENNIS K. PETERS (Senior Member, IEEE) is currently an Associate Dean of undergraduate studies and a Professor with the Department of Electrical and Computer Engineering, where he has been a member of the Faculty, since 1998. His research interests include techniques for design and verification of software and computer systems, with a particular focus on high-performance computing, real-time applications, and parallel or distributed processing. His teaching activity is primarily in the area of software, ranging from introductory programming courses to advanced topics such as real-time operating systems, and concurrent programming.

LESLEY JAMES is currently a Professor with the Department of Process Engineering, Memorial University of Newfoundland. Her research interests include sustainable offshore (Newfoundland and Labrador) oil and gas production and carbon capture, utilization, and storage (CCUS). From molecular level interactions to large field scale optimization, the challenge is to recognize the right tools given the uncertainty of the question/answer. AI/ML is proving beneficial in analyzing the complex data for the energy industry. She has led numerous large multidisciplinary research and development projects related to offshore energy. She has been awarded the Society of Petroleum Engineer's Distinguished Achievement Award for Faculty.



RAY GOSINE received the bachelor's degree in electrical engineering from the Memorial University of Newfoundland, Canada, and the Ph.D. degree in robotics from Cambridge University, U.K. He held teaching and research positions at Cambridge University, The University of British Columbia (UBC), Memorial University of Newfoundland, and the University of Toronto. These appointments included an NSERC Jr. Chair in Industrial Automation at UBC (supported by B. C. Packers) and the J. I. Clark Chair of Intelligent Systems for Operations in Harsh Environments at Memorial University (supported by C-CORE).